Accessing Federated Cloud Services with Kerberos after Network Authentication

Alejandro Abad-Carrascosa, Alejandro Pérez-Méndez, Rafael Marín-López, and Gabriel López-Millán
Dept. Information and Communications Engineering (DIIC)
University of Murcia, 30100, Spain

***Abstract*—The growth of public and private cloud services, and the proliferation of open source cloud solutions is undeniable. At the same time, new security features are being developed to provide a stronger confidence of end users and to improve the user experience. Examples of these features are advanced/extensible authentication mechanisms, Single Sign-On (SSO) and Identity Federation. This work introduces a novel approach to provide these features to cloud services, by taking advantage of well-known and widely deployed protocols. On the one hand, Kerberos, providing an advanced authentication framework and a native SSO mechanism. On the other, the AAA protocols (such as RADIUS or Diameter) providing widely deployed network access identity federations, such as eduroam. The proposed solution allows the end user to bootstrap fresh valid authentication credentials to access cloud services in a visited organization from a previous network authentication process, backed up by a AAA identity federation. Moreover, it does not imply modifications to the current standards. A proof-of-concept has been been implemented to validate the feasibility of this proposal.**

***Keywords*—federated, cloud, authentication, Kerberos, network access, OpenStack**

I. Introduction

With the rapid growth of the Internet, the deployment of cloud services have experimented a great demand [1]. As many other Internet services, allowing access to only authenticated and authorized users to cloud services is the vital importance for the Cloud Service Providers (CSPs). In parallel, the concept of Identity Federation [2] has become very relevant nowadays. Integrated with cloud services, it enables an end user, subscribed with a particular organization, the Identity Provider (IdP), to access the cloud services provided by others CSPs belonging to the same federation. This is possible thanks to a successful authentication and authorization process with the end user's credentials provided by the IdP.

The integration of both paradigms has raised special interest [3] since it provides federated access to cloud services anytime and anywhere in the federation, so that CSPs can expand the usage of its resources and services. Indeed, we can already find initiatives integrating Web-based federations (i.g. [4]) based on technologies such as SAML [5] or OAuthv2 [6]. Recently, the Moonshot/ABFAB technology [7] is being used to provide federated access to any kind of Internet services (in particular cloud services [8]) by means of existing AAA-based federations. This type of federations is typically deployed for controlling and federating the access to network access service (i.e. eduroam [9]).

Additionally, cloud services have also considered the usage of Kerberos [10] to perform access control to their cloud services [11]. The reason is that Kerberos is a reputable three-party protocol for authentication and key distribution widely used in application services, such as SSH, HTTP, etc. Its success is founded on its high level of security, Single Sign-on (SSO) capabilities and federation support, by means the so-called *Kerberos cross-realm* mechanism. Nevertheless, the deployment of Kerberos cross-realm infrastructures has been scarce. Thus, its usage has been limited to control the access of local end users registered in the CSP's domain.

Some alternatives [12]–[14] have been proposed to solve this drawback. The idea is to allow obtaining the benefits of Kerberos without deploying an alternative cross-realm infrastructure but using a more common and widely used federation substrate (i.e. AAA-based federation). However, these solutions add or inflict changes in several entities, and require new amendments to standards and related implementations. In particular, FedKERB [12] implies modifying the *Key Distribution Center* (KDC), which is the central entity in charge of distributing key material, to implement a pre-authentication mechanism for the integration with a AAA infrastructure. PanaKERB [13] implies adding a new protocol (PANA) to bootstrap Kerberos credentials in the CSP domain. Finally, EduKERB [14] uses the network access authentication to bootstrap Kerberos credentials. However, it modifies the KDC to consume a non-standardized token obtained during network access authentication.

This paper presents a simple but novel *cross-layer* solution that allows the end user to leverage the federated AAA-based authentication and authorization for the network access service to bootstrap Kerberos credentials (i.e. username and password) in order to access any deployed cloud service in the CSP's domain. Unlike EduKERB, our solution does not change any standard and involve low implementation efforts providing a neat solution keeping the main advantage: using Kerberos for federated access to cloud services by using existing AAA federations without deploying Kerberos cross-realm infrastructures.

The rest of the paper is organized as follows. Section II introduces the background technologies. Section III gives the details of our solution, explaining the architecture and message flow. In Section IV, we describe a *proof-of-concept* developed to test our idea with cloud services. Finally, section V concludes the manuscript and gives future directions.

II. Background

A. Kerberos

Kerberos [10] is a secure protocol for authentication and key management over insecure networks. It allows a *Client* (end user) to establish a dynamic trust relationship with an *Application Server* (AppS). The process is grounded on the exchange of the so-called *tickets*, that allows establishing symmetric keys between the Client and AppS with the involvement a *Key Distribution Center* (KDC) located at the service provider. These keys are called *reply key* (Client↔KDC), and *service key* (AppS↔KDC).

Kerberos involves three steps: 1) a *KRB_AS_REQ/REP* exchange that allows the Client to request a *Ticket Granting Ticket* (TGT) from the KDC. This ticket can only be processed using the Client's *reply key*; 2) a *KRB_TGS_REQ/REP* exchange executed between the Client and the KDC. It allows the Client to request a *Service Ticket* (ST) for the target AppS. The TGT is used by the Client as a proof of identity in this exchange. Finally, 3) the Client starts a *KRB_AP_REQ/REP* exchange with the AppS by using the ST. Once the AppS has verified the ST, the Client is authenticated and both share a symmetric key to protect application protocol traffic.

B. AAA/EAP

Authentication, Authorization, and Accounting (AAA) infrastructures provide a generic framework for access control in federated environments [15]. Although they were conceived to support any type of application services, they have been mainly used for the network access service so far. In AAA, an *end user*, subscribed to an organization (*home organization*), tries to access to a service (*specific equipment*) deployed by another organization (*service provider*). AAA defines how the information is conveyed from the *specific equipment* to the *home organization*, and vice-versa, in order to complete the access control functionality. This exchange involves the use of an AAA protocol (e.g. RADIUS [16] or Diameter [17]), executed between the so-called *AAA servers* deployed on each organization of the federation. Information exchange between *end user* and *specific equipment* is out of the framework.

AAA does not impose any particular authentication mechanism. However, the *Extensible Authentication Protocol* (EAP) [18] has become the most popular, as it allows using an extensible set of mechanisms and technologies without requiring the modification of the AAA protocol. This is achieved through the so-called *EAP methods* (EAP-TLS, EAP-TTLS, etc.). Besides, it adapts perfectly to the federated nature of AAA. Typically, EAP is executed between an *EAP peer* (end user) and an *EAP server* (AAA server at home organization), through an *EAP authenticator* (specific equipment). After a successful authentication, some EAP methods generate keying material [19]. In particular, the *Master Session Key* (MSK) is used to establish a security association between the *EAP authenticator*.

C. OpenStack

OpenStack [20] is an open-source cloud computing solution founded by *Rackspace* and *NASA*, developed using Python. The project aims to provide a valid solution for both, private and public clouds, focusing on the quality and extensibility of the implementation. Each one of the OpenStack's components is focusing on a part of the cloud functionality. These are *Compute* (virtual machines management), *Storage* (object and block storage), *Networking* (network and IP management), and *Dashboard* (graphical interface). They communicate through a set of well-defined *Application Programming Interfaces* (APIs). Besides, there is a set of common services making easier the operation of the cloud.

One of these services is the *Keystone*, which acts as a central authentication system for the rest of components. In particular, when an end user wants to access a specific service, she engages into an authentication process with the Keystone. The design of the Keystone allows the use of different authentication backends (e.g. LDAP, Kerberos [21], etc.). Once the authentication has been completed, the end user is supplied with a generic *unscoped token*, which can be used to request a *scoped token* for any particular cloud service and tenant (project) the end user wants. Additionally, the *Keystone* provides a rule-based authorization engine, to control who is entitled to access each particular resource.

III. Architecture and Operation Design

A. Example use case

Alice belongs to an organization called `home.org`. This organization is connected to other organizations through an AAA-based federation, by establishing the necessary trust relationships. Alice has moved to one of those member organizations called `visited.org`. This organization provides federated network access service (*network access provider*) via an AAA-based federation. Moreover, it has deployed Kerberos to control the access to its cloud services (*cloud service provider*). Alice turns on her laptop and tries to access the network, but as she has not been authenticated yet, the access is denied. Therefore, Alice makes use of the long term credentials provided by `home.org`, in order to authenticate herself against the visited organization.

Once Alice is connected to the network, she decides to access the cloud service provided by `visited.org`. She is not a member of the visited organization so she is not registered in the local KDC's database. Thus, she would need to manually contact the system administrator in order to request the creation of a set of credentials (i.e. username and password) in `visited.org`'s KDC. Similarly, those credentials must be manually removed when Alice abandons the visited organization. This procedure presents difficulties for both users and system administrators and it would be beneficial for both of them if Alice could bootstrap Kerberos credentials in an automated way. It is worth noting that this is a typical scenario we may find in networks such as eduroam.

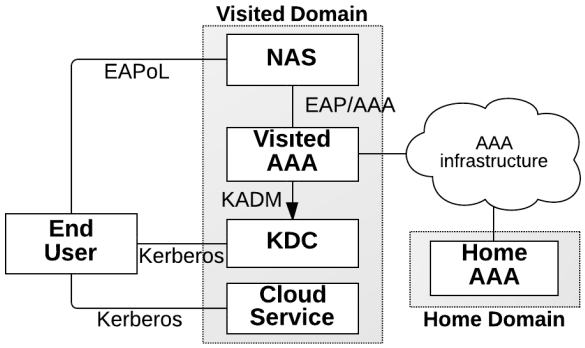

Fig. 1. Proposed Architecture

B. Objectives and Requirements

To cover the aforementioned gap, the aim of the proposed architecture is to achieve the following objectives:

1) To allow the access to federated end users to the cloud services offered by visited organizations belonging to a AAA-based identity federation. End users must only provide their long term credentials shared with their home organization.
2) To control the access to cloud services using the Kerberos authentication mechanism. Although Kerberos does already support federated access by the deployment of Kerberos *cross-realm*, it has some recognized issues [22]. Thus, federation substrate is provided by a AAA infrastructure.
3) To offer *Single Sign-On*. Users only have to introduce their long term credentials during network access authentication. After this, access to the visited organization's cloud services is granted.

This work also imposes the following requirements:

- The visited organization must authenticate the users with EAP through the AAA infrastructure before granting them access to the network.
- A Kerberos-based infrastructure is available at the visited organization to control the access to cloud services.

C. Components

Fig. 1 depicts the proposed architecture. The main entities are the following:

End user (EU). It is the entity accessing a cloud service provided by `visited.org` (Alice). The EU makes use of her `home.org`'s credentials to authenticate with EAP through the AAA infrastructure, in order to gain access to the network service. After that, she makes use of the standard Kerberos mechanism to access `visited.org`'s cloud service.

Home AAA server (H-AAA). This entity is able to verify the EU's credentials, acting as the EAP server. When the authentication successfully ends, a shared secret with the EU (i.e. MSK) is derived and delivered to `visited.org` through the AAA infrastructure.

Visited AAA server (V-AAA). It is contacted by the network access server (NAS) to transport the EAP packets from the EU to the H-AAA, and vice versa. When a EAP negotiation is accomplished, this entity issues dynamically Kerberos' session credentials (i.e. username and password) using the key material derived from the EAP negotiation. V-AAA enforces them into the end users' KDC database.

Network Access Server (NAS). This is the service equipment deployed by the visited organization in order to provide network access, such as an IEEE 802.11 [23] access point.

KDC. This entity controls the access to the cloud services in `visited.org`, authenticates the EU and provide her with Kerberos tickets to access the cloud services. The session credentials derived by the V-AAA are distributed to this entity, and used to perform the EU's authentication process.

Cloud Service (CS). This element provides some kind of cloud service to EUs. Access control is managed by Kerberos.

D. Work flow

The aforementioned entities interact in four different phases as depicted in Fig. 2, which are explained in detail below.

Phase 1: Network access authentication. In this phase, the EU is authenticated against the NAS in order to access the network. This authentication process is carried out by the EAP protocol through the AAA infrastructure. This solution does not modify the current EAP protocol at all. As a result of this operation, fresh key material (MSK) is derived and distributed between the V-AAA and NAS.

The process is triggered when the EU wants to access the `visited.org`'s network. She communicates with the NAS by using the available access technology (e.g. 802.11). This NAS is going to act as a bridge between the EU and the rest of the AAA infrastructure during the EAP negotiation. When this channel has been established, the NAS starts the EAP negotiation soliciting the identity of the EU with an *EAP Request/Identity* message *(1.1)*. In response, the EU sends an *EAP Response/Identity* message, including the users' identification, which is received by the NAS *(1.2)*. Then, the NAS encapsulates it into an AAA request message, and conveys it to the H-AAA through the V-AAA and the rest of the federated infrastructure *(1.3)*.

The H-AAA processes this request and, based on internal policies, it chooses a specific EAP method which has to be executed between the EU and the H-AAA. Then, the H-AAA sends an *EAP Request* indicating this method within a AAA reply message. This message is routed through the federated infrastructure back to `visited.org`, where it is delivered to the NAS *(1.4)*. The NAS extracts the contained EAP message, and delivers it to the EU *(1.5)*. This process continues until the EAP authentication is completed or an error happens *(1.6)*.

If the authentication process succeeds, the H-AAA includes the generated MSK in the final AAA response message, which also carries an *EAP Success* message. This information is sent to the V-AAA *(1.7)*, which stores the MSK for later use in the next phase, before forwarding the message to the NAS *(1.8)*. Finally, the NAS extracts the *EAP Success* message and delivers it to the EU *(1.9)*. At this point, the EU can

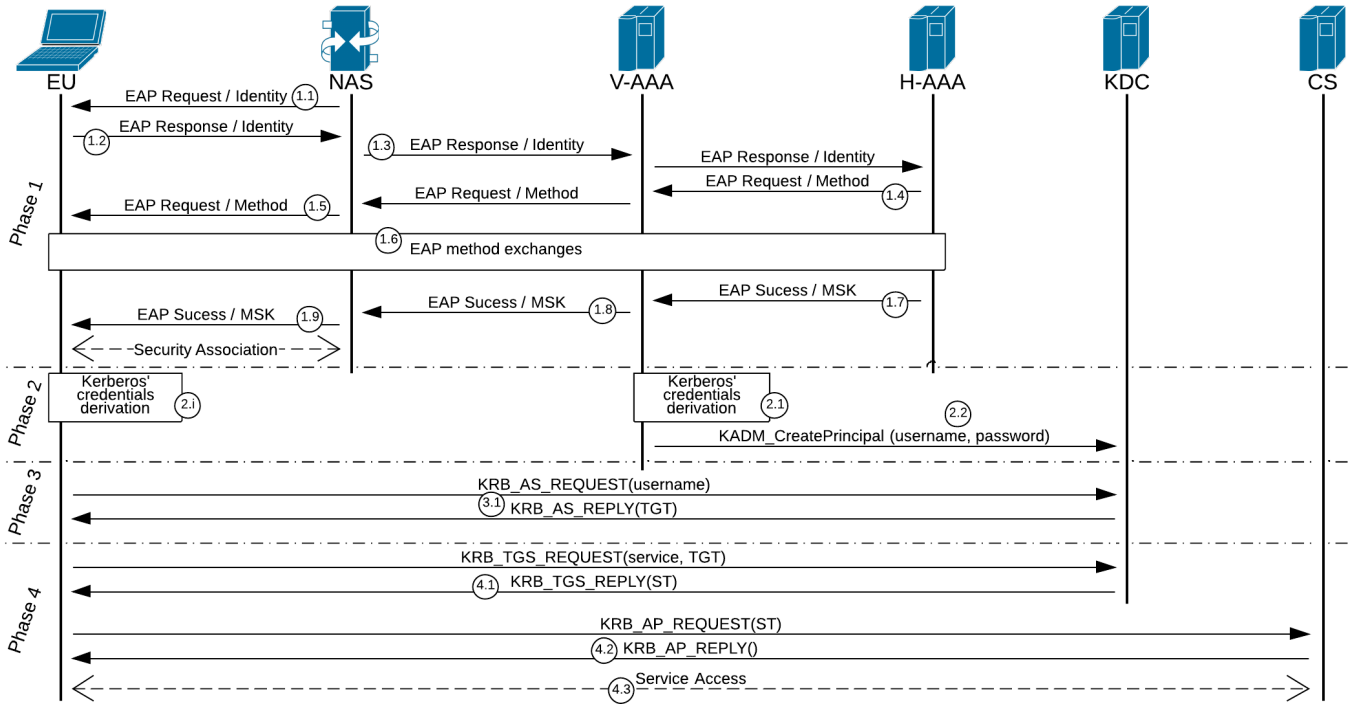

Fig. 2. Mechanism workflow

access the network, being able to obtain the necessary network configuration using any method (i.e. DHCP) .

Phase 2: Session credentials bootstrapping. During this phase, fresh session credentials for Kerberos are derived. These credentials are latter used by the EU in order to obtain a valid TGT for the visited organization. This phase is divided in two concurrent sub-phases: *phase 2a*, executed by the V-AAA and *phase 2b*, executed by the EU.

In *phase 2a*, the V-AAA makes use of the received MSK to derive a new set of Kerberos credentials for the EU. More precisely, the V-AAA derives a username and password *(2.1)*. The reason of deriving both values from the MSK is that it is the only information that is guaranteed to be shared among the EU and the V-AAA, as other information, such as the user's identifier or the long term credentials, may be unknown by the V-AAA if the executed EAP method is a tunnelled method, as EAP-TTLS or PEAP.

To derive the dynamic username and password to be installed in the KDC, we define two generic functions. The first one is used to derive the username, the second one is used to derive the user's password. Both functions take the MSK as the only input parameter. Due to Kerberos implementations expect users to introduce a textual username and password instead of binary data, we propose to transform the output of these functions into a textual representation (e.g. *Base64*). For instance, in the use case described in Section III-A, a derived Kerberos's username for Alice would look like `0123456789abcdef@VISITED.ORG`.

Once the username and password are generated, the V-AAA enforces these values into the KDC user's database *(2.2)*. The application interface with the KDC's KADM service is used by the V-AAA to make this possible.

The phase 2b starts when the EU receives the *EAP-Success* message as a result of the EAP negotiation. The EU derives then the same MSK, so she can make use of it to derive the same username and password that the V-AAA enforced into the KDC during the phase 2a. Therefore, the EU applies the same functions as the V-AAA to obtain them *(2.i)* that will allow her to authenticate against the CS by means of Kerberos.

Phase 3: Kerberos Authentication. After the enforcement of the session credentials into the KDC's database, the EU can access any application service by following the standard Kerberos protocol. Therefore, the next step is to obtain a new TGT from the KDC, by means of a *KRB_AS_REQ/REP* exchange (section II-A). This TGT is used later to obtain STs for the different cloud services offered by `visited.org`.

The EU needs to make use of her new username and password (phase 2b) that is the same that were enforced in the KDC (phase 2a). So she is able to obtain a valid TGT by means of the standard *KRB_AS_REQ/REP* exchange *(3.1)*.

Phase 4: Accessing the service. This phase is entirely based on the standard Kerberos operation, explained in Section II-A, for accessing a cloud service and, consequently, nothing needs to be modified in any of the involved entities (i.e. the KDC, the EU and the CS).

E. Security Considerations

As already stated, this solution does not inflict any changes in neither Kerberos, AAA nor EAP protocols. Therefore, their security properties remains unchanged by keeping their weak-

Component	Software	Version
CS	*OpenStack* [20] (GSS-API/Krb) [25]	0.3.0.23
V-AAA / H-AAA	FreeRADIUS [26]	2.2.4
EU	wpa-supplicant [27]	0.7.3
	Swift client (GSS-API/Krb)	1.00
KDC	MIT Kerberos [28]	1.12.1

Table I
USED SOFTWARE AND VERSIONS

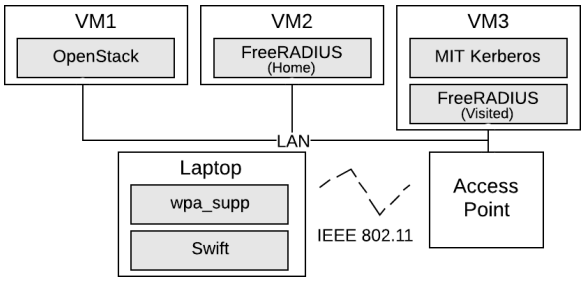

Fig. 3. Deployed Testbed

nesses and strengths. Nevertheless, there are some security considerations which impact our solution.

In terms of the Kerberos' credentials derivation process, it is highly recommended that both the username and the password derivation algorithms generate independent and irreversible outputs. Hence, an attacker owning the output from one of the functions is unable to obtain neither the MSK nor the output of the other function. We recommend to use key derivation functions such as HMAC-based (HKDF) [24].

Regarding the communication with the KADM, it is strongly recommended that the derived Kerberos' credentials are enforced using secure transport, such as TLS/SSL, and it must be assured that only authorized clients can modify the local database. Kerberos implementations are strongly encouraged to provide some kind of access control mechanism.

Regarding the validity of the derived Kerberos' credentials, it is recommended that the lifetime for these credentials is the same as the lifetime of the key material used for their derivation. It is also recommended that these credentials are considered invalid if used outside the visited organization.

IV. VALIDATION

A. Testbed

As depicted in Fig. 3, the scenario is composed by three virtual machines (VMs) using *VirtualBox* as virtualization software on top of a single host with 8GB of RAM and a quad-core *Intel Core(TM) i5-3470 @ 3.20 GHz*, a "*D-Link DWL-2100AP*" 802.11 access point (AP) with RADIUS support (SSID: "`FederatedNet`"), and a laptop with 2GB of RAM and a dual-core *AMD Athlon(TM) @ 2.30 GHz* processor. Every VM has 512MB of RAM and a single core CPU available. All the VMs and the laptop use an "Ubuntu Server 12.04 LTS" Linux distribution. Both the VMs and the AP are interconnected by means of a virtual LAN switch.

Table I lists the software used for each component. We have used existing open-source implementations with minimal modifications. In particular, to run the scenario, we have developed two software artefacts. Firstly, a FreeRADIUS module (ANSI C), called `rlm_fnasso` (*Federated Network Access Single Sign-On*) [29], whose main functionality is to obtain an available MSK within any RADIUS reply message being proxyed by the FreeRADIUS server which is running as V-AAA (VisitedRAD). This module does not modify the main core of FreeRADIUS. Once the MSK has been obtained, the module performs the derivation of the username and password for the EU, and enforces them into the KDC's database by using the *kadmin* tool; Secondly, an Unix daemon (ANSI C), called `fssod` (*Federated Single Sign-On daemon*) [29], whose main functionality is to obtain available MSKs at the EU by contacting the *wpa-supplicant* through its control interface. Once the MSK is obtained, it derives the username and password for the EU, and contacts the KDC to obtain a TGT.

B. Scenario execution

At the EU, we have configured the *wpa_supplicant* to connect to the IEEE 802.11 network with ESSID "`FederatedNet`" using EAP-TTLS. We also supplies a set of long term user's credentials registered in HomeRAD. Furthermore, when the EU's wireless interface detects the "`FederatedNet`" ESSID, it begins with the authentication process in order to gain network access.

```
user@laptop:~$ klist
(1)klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_1002)
(2)user@laptop:~$ wpa_supplicant -B -c/etc/wpa_supplicant.conf -iwlan0
user@laptop:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1002
(3)Default principal: d31092e9aa01@VISITED.ORG

Valid starting     Expires            Service principal
23/07/14 19:07:39  23/07/14 20:07:35  krbtgt/VISITED.ORG@VISITED.ORG
user@laptop:~$ swift --kerberos \
-A http://keystone.visited.org:5000/v3/auth/tokens list
(4)Authentication successful using "d31092e9aa01@VISITED.ORG" kerberos identity
TestContainer2
TestContainer
```

Fig. 4. Console output

Fig. 4 shows an execution of the deployed testbed from the point of view of the EU. Initially, the Kerberos credentials' cache is empty *(1)*. At the laptop, we trigger a network connection using *wpa_supplicant (2)* to connect to the ESSID "`FederatedNet`". As the supplied credentials are valid, EAP finishes successfully and the laptop gains network access. Then, the `rlm_fnasso` module running at the VisitedRAD detects this event and triggers the derivation and enforcement of the new username/password into the KDC's database. Concurrently, the `fssod` daemon also detects the connection event, triggering the derivation of the session credentials and using them to contact the KDC in order to obtain the valid TGT for the visited domain. Furthermore, after the network connection is completed, the user can observe that a new TGT has been automatically stored in the credentials' cache for an

user named "d31092e9aa01@VISITED.ORG" *(3)*. Using this TGT we run *Swift* client in order to authenticate against *KeyStone/OpenStack* and to execute a simple list command. Due to the TGT obtained during the network authentication process, the end user is able to access a specific cloud service without having to introduce any further credentials *(4)*.

V. Conclusion

This paper introduces a federated authentication mechanism that integrates cloud services, Kerberos and AAA. On the one hand, the end user is registered in a home organization within a AAA-based federation used to control the access to the network access service in a visited organization. The visited organization also provides cloud services.

With a single authentication process, performed to obtain network access service through a AAA-based federation, our solution allows the end user to request access to the set of kerberized cloud services provided by the visited organization. This has been possible by bootstrapping a dynamic Kerberos username and password with the key material obtained after a successful network access authentication. Both username and password are enforced in the KDC by the visited AAA server during the network access authentication. Thus, the end user is registered in the visited organization dynamically, reducing management efforts for federated end users. Moreover, the end user requesting access to a cloud service in the visited organization does not require re-introducing her long term credentials. The simplicity of this solution makes its adoption easier, being specially interesting for organizations which have already deployed a Kerberos infrastructure.

As future work, we will consider the integration with other identity federations technologies, such as ABFAB/Moonshot, and the addition of advanced authorization mechanisms based on SAML and XACML. We will also analyze the case where the KDC is only located at the home organization.

Acknowledgment

This work is supported by the project 16822 (CLASSe) - MULTI-GIGABIT EUROPEAN RESEARCH AND EDUCATION NETWORK AND ASSOCIATED SERVICES. Authors thank the Funding Program for Research Groups of Excellence with code 04552/GERM/06 granted by the Fundación Séneca.

References

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," in *National Institute of Standards and Technology Special Publication 800-145*, 2011.

[2] A. Perez, F. Pereniguez, R. Marin, G. Lopez, and J. Howlet, "Identity Federations Beyond the Web: A survey," in *IEEE Communications Surveys & Tutorials*, May 2014, pp. 891–895.

[3] V. Casola, M. Rak, and U. Villano, "Identity federation in cloud computing," in *2010 Sixth International Conference on Information Assurance and Security (IAS)*, Aug. 2010.

[4] D. Chadwick, K. W. S. Siu, C. Lee, Y. Fouillat, and D. Germonville, "Adding Federated Identity Management to OpenStack," *Springer Journal of Grid Computing*, pp. 3–27, 2014.

[5] S. Cantor, J. Kemp, R. Philpott, and E. M. (Eds.), "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) v2.0," Mar. 2005, oASIS.

[6] E. Hammer-Lahav, D. Recordon, and D. Hardt, "The OAuth 2.0 Authorization Protocol," RFC 6749, Internet Engineering Task Force, Oct. 2012.

[7] J. Howlett and S. Hartman, "Project Moonshot," available from: https://community.ja.net/groups/moonshot. Accessed 7 July 2014.

[8] "Cloud-ABFAB Federation Services in eduroam," available from: http://www.um.es/CLASSe. Accessed 7 July 2014.

[9] L. Florio and K. Wierenga, "Eduroam, providing mobility for roaming users," in *Proceedings of EUNIS 2005 Conf., Manchester*, 2005.

[10] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, "The Kerberos Network Authentication Service (V5)," RFC 4120, Internet Engineering Task Force, Jul. 2005.

[11] M. Hojabri and K. V. Rao, "Innovation in cloud computing: Implementation of Kerberos version5 in cloud computing in order to enhance the security issues," in *2013 International Conference on Information Communication and Embedded Systems (ICICES)*, 2013, pp. 452–456.

[12] R. Marin, F. Pereniguez, G. Lopez, and A. Perez, "Providing EAP-based Kerberos pre-authentication and advanced authorization for network federations," *Computer Standards & Interfaces*, vol. 33, no. 5, pp. 494–504, 2011.

[13] A. Perez, F. Pereniguez, R. Marin, and G. Lopez, "Out-of-band federated authentication for Kerberos based on PANA," *Elsevier Computer Communications*, vol. 36, no. 14, pp. 1527–1538, 2013.

[14] A. Perez, F. Pereniguez, R. Marin, and G. Lopez, "A cross-layer sso solution for federating access to kerberized services in the eduroam/dame network," *International Journal of Information Security*, vol. 11, no. 6, pp. 365–388, 2012.

[15] "Authentication Authorisation Accounting Architecture Research Group," available from: http://irtf.org/concluded/aaaarch. Accessed 7 July 2014.

[16] B. Aboba and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)," RFC 3579, Internet Engineering Task Force, Sep. 2003.

[17] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn, "Diameter Base Protocol," RFC 6733, Internet Engineering Task Force, Oct. 2012.

[18] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, "Extensible Authentication Protocol (EAP)," RFC 3748, Internet Engineering Task Force, Jun. 2004.

[19] B. Aboba, D. Simon, and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework," RFC 5247, Internet Engineering Task Force, Aug. 2008.

[20] "OpenStack Open Source Cloud Computing Software," available from: https://www.openstack.org/. Accessed 7 July 2014.

[21] "Victor M. Ruiz's GitHub Repository," Available from: https://github.com/VMRuiz. Accessed 31 July 2014.

[22] S. Sakane, K. Kamada, S. Zrelli, and M. Ishiyama, "Problem Statement on the Cross-Realm Operation of Kerberos," RFC 5868, Internet Engineering Task Force, May 2010.

[23] "IEEE Standard for Information Technology," *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, Jun. 2007.

[24] H. Krawczyk and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)," RFC 5869, Internet Engineering Task Force, May 2010.

[25] L. Zhu, K. Jaganathan, and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2," RFC 4121, Internet Engineering Task Force, Jul. 2005.

[26] "The FreeRADIUS Project," Available from: http://freeradius.org/. Accessed 10 June 2014.

[27] "Linux WPA/WPA2/IEEE 802.1X Supplicant," Available from: http://hostap.epitest.fi/. Accessed 10 June 2014.

[28] "MIT Kerberos," Available from: http://web.mit.edu/kerberos/. Accessed 26 April 2014.

[29] "Alejandro Abad's GitHub Repository," Available from: https://github.com/AlejandroAbad. Accessed 2 April 2014.