An Approach for Integrating Kerberized non Web-Based Services with Web-Based Identity Federations

Aleksandr Bersenev[1], Arsen Hayrapetyan, Marcus Hardt[2]
[1]Ural Federal University
Russian Federation
[2]Karlsruhe Institute of Technology
Germany
[1]Alexander.Bersenev@urfu.ru, [2]arsen.hayrapetyan@kit.edu, hardt@kit.edu

ABSTRACT: *Many identity federations are designed to be used with web browsers. This paper proposes a approach for integrating non web-based applications with web-based identity federations using Kerberos protocol. We evaluate this approach by making NFS server available for users of SAML-based identity federation of Baden-Württemberg state of Germany. We make use of LDAP-Facade software for federating non web-based services. We have modified the web-interface component of LDAP-Facade to enable the registration with kerberized services. Our approach can be used without modifications on the client side.*

Keywords: SAML, Kerberos Protocol, LDAP, Web Based Applications

Received: 10 October 2014, Revised 14 November 2014, Accepted 19 November 2014

1. Introduction

Identity federations allow users to authenticate on services hosted by external organizations with their home organization accounts. Rather than storing all identities in the same place, each organization is responsible for keeping an up to date information about its users and authenticating them, becoming the Identity Provider (IdP). Service Providers (SP) consume the information from IdPs they trust.

There are a number of technologies for establishing identity federations. Authors of [1] divide them into three groups:

- Web-based identity federations which base on standards like SAML (Security Assertion Markup Language), OpenID (Open Identity) or OAuth (Open Authorization);
- AAA-based (Authentication, Authorization and Accounting) identity federations. They use protocols like RADIUS or Diameter for data transfer. Both of those protocols support EAP (Extensible Authentication Protocol) for authentication. An example of a federation of this type is the eduroam network;
- Application-Agnostic Identity Federations, which rely on Kerberos in cross-realm operation or on middleware layers like GSS-API (Generic Security Services API) [2] or SASL (Simple Authentication and Security Layer) [3].

In addition, Grid Trust Federations are based on X.509 standard PKI and use GSS-API middleware layer for authentication ant authorization.

This paper focuses on web-based identity federations. On this type of federations the primary protocol for communications with IdPs and SPs is HTTP, often over TLS [4]. SPs usually redirect browsers to the IdP for authentication and can consume tokens from it.

However there are a lot of popular non web-based protocols, for example SSH, NFS, RDP, IMAP or SIP. Most implementations of these protocols can not use authentication against web-based identity federations out of the box yet, but instead support the Kerberos authentication protocol.

In the first part of this paper we propose an approach how to pass a federated authentication result to a service with Kerberos protocol as a mediator. Our approach does neither require modification of the Kerberos server nor of the service. Extensions on the client side will be described.

In the second part we describe one way of how this approach can be implemented in practice. We used these components:

- NFS-Ganesha as an NFS (Network File System) server implementation;
- SAML-based identity federation of Baden-Württemberg state called bwIDM;
- Modified LDAP-Facade as an implementation of an SP and its LDAP-backend as a storage for service-local data.

1.1 The Background

The Baden-Württemberg state has an identity federation called bwIDM federation[1]. It includes several dozens of IdPs from different universities and several SPs. The bwIDM federation is a subfederation of another federation, the DFN-AAI[2] – a Germany-wide identity federation for scientific institutions. DFN-AAI is based on SAML. Most of its IdPs and SPs are using Shibboleth implementation of SAML.

To add the SAML-based authentication features to non web-based services the abstract layer called LDAP-Facade was developed at Karlsruhe Institute of Technology (KIT). It consists of an LDAP-server, which is accessable for the non web-based services, and a web-interface, which authenticates users with SAML and allows them to manage data on the LDAP server. They can register for a particular service (e.g. add their entry to the LDAP), get their data, change some parts of it (e.g. local service passwords), or unregister (delete their entry from the LDAP).

LDAP-Facade is already used to federate several non webbased services in KIT. Some of them store scientific data for more than 100 000 students in Baden-Württemberg. This data can be accessed via various protocols, but no NFSv4-based access was available at the time of this publication.

NFSv4 has several features some other protocols lack:

- It is standardized and supported by all major OS;
- It is scalable because of optional parallel operation(pNFS);
- It uses traditional Unix permissions model and can use access control lists;
- It has a data encryption option.

Since Kerberos is the only protocol for user-based authentication implemented by NFS servers [5], we were looking for ways to have the LDAP-Facade work with it. In the previous version, NFS 4.0, it was required to implement two alternative mechanisms for authentication: LIPKEY and SPKM-3 [6], but they were unsupported by all major implementations and this requirement was dropped in the NFS 4.1 standard [7].

---

[1]Föderiertes Identitätsmanagement der baden-württembergischen Hochschulen, https://www.bwidm.de/, last checked: 2015-02-27

[2]DFN-AAI, https://www.aai.dfn.de/, last checked: 2015-02-27

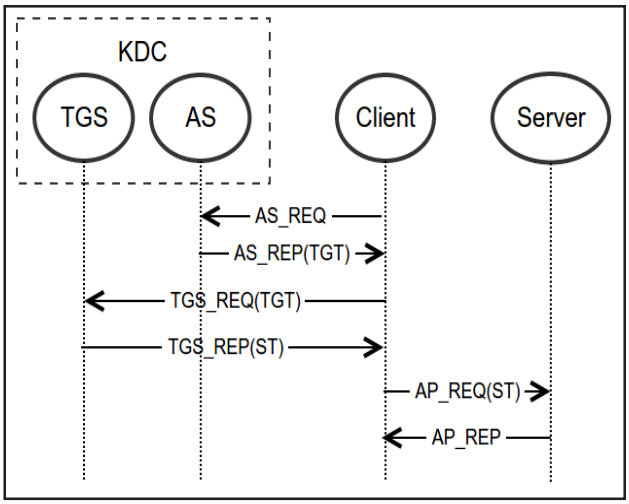

Figure 1. Typical Kerberos workflow

Extending the LDAP-Facade to Kerberos allowed us to apply the approach to additional services. We found that our approach had nothing specific to SAML and LDAP-Facade and we have extended it to the abstract web-based federation.

1.2 Kerberos Authentication Protocol

The Kerberos protocol is described in detail in [8]. It requires a trusted third party and uses symmetric key cryptography.

The KDC (Key Distribution Center) is a server part of Kerberos. It consists of two subservices: AS (Authentication Server) and TGS (Ticket-Granting Server). Tickets are used to verify the identity of the client for a particular service.

The AS is responsible for user authentication and for granting initial ticket (called Ticket Granting Ticket or TGT) which verifies the identity of a client for TGS. Authentication is based on preshared key, so AS has access to a database of keys for each client. Usually, the client derives its key from the password by using a string-to-key function. If an attacker manages to capture AS_REP, he can try to bruteforce the user password [9]. The Kerberos specification does not handle this and recommends the use of complex passwords. To protect from active attacks of this type AS can require pre-authentication before issuing the TGT.

TGS is responsible for issuing service tickets. The service ticket is encrypted by the service key. Every service has such a key, hence communication between service and Kerberos is unnecessary. Figure 1 shows the three data exchanges required to autenticate to a service: AS_REQ/AS_REP, TGS_REQ/TGS_REP and AP_REQ/AP_REP. The Kerberos standard allows multiple AS_REQ/AS_REP data exchanges, if needed.

2. Related Work

To address federated identity issues for non web-based services, the Application Bridging for Federated Access Beyond web (ABFAB) working group was established [10]. They focus on creating a design that will combine the existing protocols, such as RADIUS, EAP and SAML. The group uses GSSAPI to integrate the established mechanisms with application protocols.

The implementation of the set of standards developed by the ABFAB working group is the Moonshot Project. It requires modifications of servers, clients and an IdPs [1].

In the same time authors of [11] proposed an approach of integrating Kerberos pre-authentication with existing AAA (Authentication, Authorization and Accounting) infrastructure by using EAP. Their approach requires modification of clients, Kerberos server, IdPs and modifies the Kerberos protocol removing the “statelessness” from Kerberos’s Key Distribution Server (KDC).

Two years later the same group of authors proposed to employ an authentication mechanism independent of Kerberos [12]. They use Protocol for Carrying Authentication for Network Access (PANA) as out-of-band protocol. This enables the use of

standard Kerberos servers, but requires modified clients.

The scope of all these projects is the AAA-based identity federations. This is a big drawback for organizations, who use web-based identity federations instead, like bwIDM, because they are not designed to support such protocols and it requires a lot of organizational and development work to integrate them.

An approach to federating non web-based services in webbased identity federations was described in [13]. The authors federate services by developing access control modules, using existing security interfaces of particular services, e.g. the Pluggable Authentication Module (PAM) interface used by many unix services, for example by the SSH-server. These modules consume assertions from the IdP. This approach works with applications that are able to use PAM.

The same group of authors extended their work in [14] including the web registration application, called LDAP-Facade, and offering a generic LDAP interface to services. We use a modified LDAP-Facade to implement our approach in practice.

3. Proposed Approach

The main idea of our approach is to set up a web-tokerberos service. This service forwards successful authentication against a web-based identity federation to KDC. To do so, it sends the user her client key via the HTTPS (HTTP+TLS) protocol to protect from untrusted network. This key can be obtained either manually by browser or automatically by a key-obtaining script on the client side.

In contrast to federated Kerberos approach, which requires every user to authenticate with KDC in their home organization, KDC in our approach operates on the SP-side (which can be in external organization). The services share their keys with that KDC. After the user has got her key, she can obtain TGT from KDC with the command-line tools (e.g. kinit). Having TGT it is possible to get service tickets from TGS to authenticate.

Figure 2 shows this workflow.

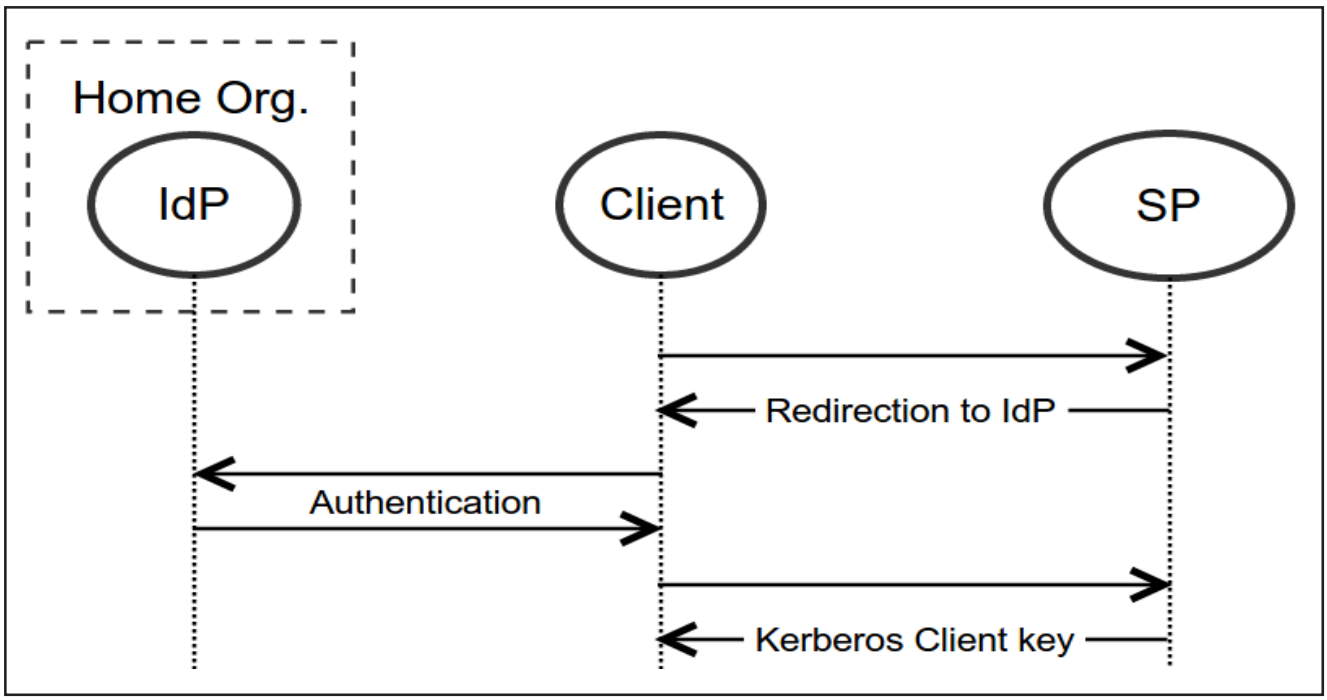

Figure 2. Before AS_REQ: User gets Kerberos client key with HTTPS

3.1 Initial Key Placement

There are several options how the client key can be generated on the web-to-kerberos service at the first time:

1) From service-local password, specified by user with the web-service. As said before, if the password is simple, the attacker,

who has the AS_REP message can try to bruteforce it. Password complexity checks should be in place.

The key sending step is unnecessary – the client can derive it from the password on his own host;

2) From the result of base64 of a long random string;

3) From the result of keyed-hash message authentication code(HMAC) [15] cryptographic mechanism with a long string as a key(shared between web-service and KDC) and user principal as a message. The KDC needs to be modified to obtain keys dynamically from the HMAC mechanism.

When the first or second option is used, KDC and webservice must have access to a common key storage. Most implementations of KDC support querying data from a key storage with an LDAP protocol.

The third option can be used without common key storage. The cost for this is the modification of a Kerberos service and difficulties on user account deprovisioning.

3.2 Converting Federated User Identifiers to Kerberos Principals

To make Kerberos Principals unique for users with the same names, but from different organizations, the Kerberos principal name should include the unique name of the IdP. It may be difficult for a user to remember it, so some mapping from IdP names to short name prefixes proved to be useful.

3.3 Authorization and Deprovisioning

The Kerberos standard [8] does not recommend services to base its authorization only on ticket issuance because this check can be bypassed if the application uses other authentication methods. If it is necessary, the Kerberos server can be modified to check client authorization before issuing each service ticket. After the user account is deprovisioned and has her record removed from the KDC database, she is still able to authenticate with tickets she already obtained. To minimize negative effects from it, the maximum service ticket lifetime should be set to a reasonably low value.

The authorization should be implemented on the application level. Before the user is let in, the application should check the user against her identity federation. If the identity federation does not support such types of checks, the application can make a request to SP web-service by using a common protocol, like LDAP.

In our case, checks are done by the LDAP-Facade and all provided attributes are mapped to a group-id that can be used for authorization.

3.4 Security Considerations

Historically Kerberos protocol version 4 used only DES (Data Encryption Standard) encryption, which has an effective key size of 56 bits. This makes the whole keyspace prone to brute-force attacks on modern hardware. In Kerberos version 5 the support of other ciphers was added and DES was deprecated by [16]. The DES encryption support should be turned off on both server and client.

To protect from replay attacks, a secure clock synchronization protocol should be used where possible. LDAP and HTTP protocols should be used only with TLS.

4. Evaluation

To evaluate deployment efforts and operation of our approach in practice, we implemented it using Ganesha-NFS as service, a modified LDAP-Facade as an SP and bwIDM as a SAML-based identity federation.

4.1 LDAP-Facade

Two major LDAP-Facade components are a web-application and an LDAP-server. The web-application is an SP in terms of SAML. It allows a user to log in with her home IdP and register for one or several services. It keeps a database of registered users for each service which it makes available to services via LDAP protocol. The LDAP directory has a tree structure, the serivces' data about users and groups is located in non-intersecting subtrees (partitions) of it. LDAP-Facade enables to store service-local data (service-local password, local name, unique identifiers, primary group identifier and so on) in the attributes of the user entry.

The authors of LDAP-Facade use specially-configured ApacheDS open source implementation of LDAP-Server, which has a Kerberos server embedded. This Kerberos server stores KDC-specific data (principal names, client and service keys, key version numbers) in the entries of krb5KDCEntry class (which is subclass of krb5Principal). The Kerberos server is tightly integrated with the LDAP part of ApacheDS, for example, if the user entry has a krb5PrincipalName attribute, her keys are regenerated after each password change.

Our first modification of LDAP-Facade was made to assign krb5KDCEntry class to the freshly-created user accounts. Mandatory attributes (krb5KeyVersionNumber and krb5PrincipalName) are also filled.

ApacheDS stores Kerberos keys in the krb5Key attributes of entries which belong to krb5KDCEntry class. Each krb5Key attribute consists of a key type and a key itself and is encoded using ASN.1 (Abstract Syntax Notation One) standard.

The second modification of LDAP-Facade allowed user to download her Kerberos client key from the web-service. The web-service redirects the user to her home IdP for authentication following standard SAML Web Browser SSO profile [17].

4.2 NFS Specifics

We use Ganesha-NFS as NFS server, because it allows to export clustered GPFS (General Parallel File System) with pNFS protocol. Furthermore it works in userspace, which allows debuging and patching it more easily.

The setup with the NFS server implemented in Linux kernel 3.19.0 on Gentoo Linux was also tested.

Both implementations support only Kerberos as user-based authentication. They can operate in three modes (security flavors):
1) krb5 requires user to prove her identity cryptographically on each RPC request. Identity of server is also verified;
2) krb5i is like krb5, but additionally protects integrity of the data on each RPC request;
3) krb5p is like krb5i, but prevents any data exposure by encrypting it.

We use the NFS client as implemented in the Linux kernel. It requires a userspace process rpc.gssd to be launched. By default, to mount NFS, the principal of the client machine is used. Since our approach does not provide any means to get principals for machines, this behavior should be turned off using the '-n' option of rpc.gssd process.

NFS internally uses numerical user and group ids, but not principal names. If information about files owner and group is important, some mechanism should be set up on the client to map numerical ids to the readable names and vice versa. This can be done statically by using the generic idmap mechanism. More convenient way is to set Linux up to use LDAP for such mapping with the Name Service Switch mechanism.

Since NFS has a lot of specifics, it may be inconvenient to require a user to mount NFS shares manually. To make NFS mounting easier for the user we developed a shell-script. This script executes the following steps:

1) Checks if required software is installed;

2) Checks that rpc.gssd process is launched with '-n' option;

3) Asks user for her authentication credentials at IdP;

4) Authenticates on IdP and uses the data from it to log in to SP;

5) Downloads Kerberos client-key from the SP in a keytab format;

6) Uses Kerberos client-key to obtain TGT;

7) Mounts NFS share in the specified directory.

Steps 3-5 are optional if user already has its Kerberos client key. To avoid storing Kerberos client key on physical storage, virtual file systems like tmpfs can be used.

To access the share, each Linux user on the client side must have a valid ticket. Usually, only superuser can make a mount, so every other user should obtain the ticket in order to use mounted share.

4.3 Other Tested Setups

The ApacheDS's implementation of KDC is able to serve only one Kerberos realm at a time. In the case we have distinct set of users for each service, the other implementation of the KDC may be used. We tested our setup with MIT krb5 server. It stores all client keys in the LDAP encrypted with master-key. To be able to retrieve and decrypt keys the masterkey should be shared between KDC and web-serivce. Using distinct Kerberos servers makes deployment more difficult. We successfully tried our approach also with SSH and MongoDB (Enterprise server and a standard client).

5. Conclusion and Future Work

In this paper we presented a general approach to integrate web-based identity federations with non web-based services that support Kerberos autentication protocol. It does not require service or client modification or modification of standard protocols. The components that should be set up are: webservice, KDC service and a common database. We evaluated this approach, using existing bwIDM identity federation and modified LDAP-Facade software as an implementation of aforementioned components. For our case deployment efforts were minimal. As non-web based kerberized services we have chosen NFS, SSH and MongoDB.

In the future we plan to test our approach with more kerberized services. We will also work on the improvement of commandline user tools for the management of their Kerberos credentials.

References

[1] Pérez-Méndez, A., Pereñíguez-García, F., Marín-López, R., López- Millán, G., Howlett, J. (2014). Identity federations beyond the web: A survey, *IEEE Communications Surveys and Tutorials*, 16 (4), p. 2125– 2141.

[2] Linn, J. (2000). Generic security service application program interface version 2, update 1, Internet Requests for Comments, RFC Editor, RFC 2743, January 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2743.txt

[3] Melnikov, A., Zeilenga, K. (2006). Simple authentication and security layer (sasl), *Internet Requests for Comments*, RFC Editor, RFC 4422, June 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4422.txt

[4] Dierks, T., Rescorla, E. (2008). The transport layer security (tls) protocol version 1.2," Internet Requests for Comments, RFC Editor, RFC 5246, August. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[5] Adamson, W., Williams, N. (2014). Nfsv4 multi-domain fedfs requirements, Working Draft, IETF Secretariat, Internet-Draft draft-adamsonnfsv4- multi-domain-federated-fs-reqs-05.

[6] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., Noveck, D. (2003). Network File System (NFS) version 4 Protocol, Internet Requests for Comments, RFC Editor, RFC 3530, April. [Online]. Available: http://www.ietf.org/rfc/rfc3530.txt

[7] Shepler, S., Eisler, M., Noveck, D. (2010). Network File System (NFS) Version 4 Minor Version 1 Protocol, Internet Requests for Comments, RFC Editor, RFC 5661, January 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5661.txt

[8] Neuman, C., Yu, T., Hartman, S., Raeburn, K. (2005). The kerberos network authentication service (v5), Internet Requests for Comments, RFC Editor, RFC 4120, July. [Online]. Available: http: //www.ietf.org/rfc/rfc4120.txt

[9] Wu, T. D. (1999). A real-world analysis of kerberos password security. in NDSS. The Internet Society, [Online]. Available: http: //dblp.uni-trier.de/db/conf/ndss/ndss1999.html#Wu99

[10] Smith, R. (2012). Application bridging for federated access beyond web (abfab) use cases, Working Draft, IETF Secretariat, Internet-Draft draft-ietfabfab- usecases-05.

[11] Marín-López, R., Pereñíguez, F., López, G., Pérez-Méndez, A. (2011). Providing eap-based kerberos pre-authentication and advanced authorization for network federations, *Computer Standards and Interfaces*, 33 (5), p. 494–504.

[12] Pérez-Méndez, A., Pereñíguez-García, F., Marín-López, R., López- Millán, G. (2013). Out-of-band federated authentication for kerberos based on pana, *Computer Communications*, 36 (14), p. 1527–1538.

[13] Köhler, J., Labitzke, S., Simon, M., Nussbaumer, M., Hartenstein, H. (2012). Facius: *An easy-to-deploy saml-based approach to federate non webbased services*, p. 557–564.

[14] Köhler, J., Simon, M., Nussbaumer, M., Hartenstein, H. (2013).Federating hpc access via saml: Towards a plug-and-play solution, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*), 7905 LNCS, p. 462–473.

[15] Krawczyk, H., Bellare, M., Canetti, R. (1997). Hmac: Keyed-hashing for message authentication, Internet Requests for Comments, RFC Editor, RFC 2104, February. [Online]. Available: http://www.ietf.org/rfc/ rfc2104.txt

[16] Astrand, L. H., Yu, T. (2012). Deprecate des, rc4-hmac-exp, and other weak cryptographic algorithms in kerberos," Internet Requests for Comments, RFC Editor, RFC 6649, July. [Online]. Available: http://www.ietf.org/rfc/rfc6649.txt

[17] Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra, P. Philpott, R., Maler, E. (2005). *Profiles for the oasis security assertion markup language (saml)* v2.0, OASIS Standard, March 2005. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf