

TLS-Federation

very Initial Draft: please, do not distribute

version 0.1 (June 11, 07) – very first version by Bud—no review yet by other authors

Bud P. Bruegger <bud@comune.grosseto.it>, Comune di Grosseto, Italy

Detlef Hühnlein <Detlef.Huehnlein@secunet.com>, Secunet, Germany

other co-author TBA

Abstract

Introduction

Transport Layer Security (TLS) [XXX RFC XXX], formerly known as Secure Socket Layer (SSL), is the universally accepted work horse for Internet security and strong authentication. It is ubiquitously used for adding security and confidentiality to a wide variety of applications ranging from HTTP to Virtual Private Networks.

The TLS standard is very mature and highly stable; technology that supports TLS has been ubiquitous for years; and the security of the protocol and its implementations has been so widely reviewed and tested that it is likely one of the most secure choices to date.

Optionally, TLS also incorporates X.509 certificate-based client authentication that results in highly secure authenticated secure channels and sessions. This functionality is typically used for secure, authenticated access to on-line services. The major browsers support both file based “soft” X.509 credentials and, using a specific middleware, smart card and other token-based X.509 credentials such as eID cards. There are standard APIs for the necessary middleware (CSP for Internet Explorer and PKCS#11 for Mozilla family browsers, respectively). Similarly, all major web servers support TLS with client-certificate-authentication.

Several national eID projects promote TLS as the basic authentication mechanism for the access of web-based services. The probably most visible example for this is the Belgian Reverse Proxy [Stern2005: http://www.belgium.be/zip/eid_authentication_proxy_fr.html] that is based on the Apache web server. The current European national eID infrastructures are typically designed to support TLS-based authentication, while they currently lack key components (such as Identity Providers) necessary for other authentication protocols such as those of Liberty Alliance ID FF [XXX] or WS-* [XXX].

The use of TLS authentication is usually thought of being limited to X.509 credentials such as file- or smart-card-based certificates/key pairs. This makes TLS seemingly unsuited for contexts that require federation and a free choice of credential technology including username/password or non-X.509-based smart card credentials such as the Austrian Citizen Card [XXX].

This paper reasons that TLS can be seen as an authentication protocol that is conceptually independent from the actual credential technology that is used. For this purpose, it foresees a dynamic translation of an arbitrary credential into an X.509 credential that is then suited to be used in the TLS authentication. This translation process is directly comparable to the issuance of an authentication assertion by an Identity Provider in the Liberty Alliance ID FF approach. Translation is only necessary for credentials that are not already X.509.

The paper illustrates how this TSL-Federation is already completely covered by highly mature IETF standards and that it uses the very same architectural components that are standard in the X.509 domain and are ubiquitously available on the market.

TLS federation is thus not a new standard or a new technology; it is much rather an unusual use of existing standards and technology. The only new thing is that the browser middleware interacts in standard ways with an Identity Provider in the case of non-X.509-credentials, instead of always interfacing to a local X.509 token such as a smartcard.

The paper is organized as follows: XXX

The Concept of Federation

The term “federation” is commonly used in the context of concrete technical solutions such as those by Liberty Alliance or WS-*. In order to understand TLS-Federation, it is important to make the key concepts behind these technical solutions explicit; this is a pre-requisite for reasoning that TLS actually can be used as a federated solution and it also is the basis for comparison of different competing federation solutions.

A conceptual focus is also important to break out of domain-limited thinking. For example, the main federation solutions to date have been designed with a typical focus on username/password that has generally been the predominant credential type. Understanding how the situation changes when applying federation to an eID context--where username/password is either absent or at least a rare exception—requires a much more conceptual and domain-neutral thinking.

The following sections describe the key concepts of federation. They can actually be seen as the needs or requirements for which federation solutions have been designed and for which they are in use.

Federation of Trust

One key requirement for federation is that the identity management system must be able to support multiple authorities who issue credential and certify personal data. This requires trust relationships.

A so called “relying party”, usually a service provider, must trust in the certificates/assertions/claims issued by a so called “identity provider”. Identity provider is a technology specific term that is used in the context of Liberty Alliance and WS-*; the concept of Certification Authority used in the context of X.509 technology is conceptually equivalent.

Liberty Alliance uses the term “circle of trust” to illustrate the trust relationship among a group of distinct and independent players.

Different federation solutions foresee different formats for the data to be signed and use different names for the signing authorities, but conceptually there is a very high degree of equivalence. For example, in the Liberty Alliance approach, an *Identity Provider* signs a SAML *assertion*; in the WS-* approach, and *Identity Provider* signs a *claim*; and in TLS-Federation, a *Certification Authority* signs a *certificate*. But in all cases, the signing authority is identified by an X.509 certificate and PKI approaches are used to establish trust.

Federation of trust makes it possible to use credentials issued by different trusted parties. This is most important in a username/password approach where the validation of credentials requires some central directory that manages username/password information. Obviously, in many real-world deployment scenarios, a single credential repository would be inconceivable. Some federation solutions were designed to provide additional “plumbing” that makes it possible to convey trust even in the scenario of distributed credential repositories. This additional plumbing typically takes the form of adding Identity Providers who vouch for the validation of credentials by presenting their trusted X.509 certificate together with a signature.

It is evident that trust-federation is very different when looking at X.509 credentials. In this case, the credential itself already is autonomous without the need of neither a credential repository nor an Identity Provider who vouches for the credential. Instead, the credential--with its key-pair that is used to sign a challenge--is totally autonomous in the process of validation of the credential; and it already contains an X.509 certificate that is signed by a trusted authority.

Thinking of the mechanisms for verifying the revocation status of X.509 credentials as limitation of autonomy is misleading. Certificate revocation lists and OCSP responders are really required by the digital signature as used in any of the federation solutions, not by the X.509 credential in itself. It is conceptually equivalent whether the signature applies to an X.509 certificate, a SAML assertion, or a WS-* claim; in all cases the signature verification requires that the revocation status of all certificates in the trust chain be checked.

Independence of Credential Technology

Federation is also used to achieve interoperability across different credential types in a unified identity management system. A prime example for this is the European eID Interoperability Framework that is needed as a consequence of the Manchester Declaration [xxx]. Here, following the principle of subsidiarity, European Member States have full autonomy in their choice of credential technology used in their eIDs; and while X.509 is by far the most common technology chosen, there are also smartcard-based eIDs that use proprietary authentication mechanisms¹, and countries who use username/password credentials.

For practical reasons, a unified identity management system typically agree on a single credential type that is accepted by relying parties; the support of multiple credential types that may change over time by a potentially very large number of service providers is often prohibitively complex to manage.

For this reason, all federation solutions decide on a single credential types to be used to authenticate sessions (i.e., a session credential). Liberty Alliance uses SAML assertions as session credentials; WS-* uses claims; TLS-Federation uses X.509 certificates. In each case, relying parties expect this and only this session credential type.

Federation solutions further provide a mechanism that allows to translate the original arbitrary credential in the agreed-on session credential. This translation is the responsibility of an Identity Provider. The example of the European eID Interoperability scenario further illustrates how such credential translation takes place. To overcome the complexity of a potentially different choice of eID technology by every Member State, every country sets up a national Identity Provider that translates from the national to the framework credential. Typically, users are asked to authenticate with their eID to the national Identity Provider who validates the eID and then issues a session credential.

While in Liberty Alliance and WS-*, the chosen session credential is thought of being conceptually different from an user-owned (eID) credential, TLS-Federation takes the approach of purposely choosing the most common user-owned credential type as session credential. The motivation for this approach is a significant simplification since the “additional plumbing” is only necessary in the relatively rare cases where a different user-credential type has been chosen.

Single-Sign-On

A third main requirements for federation solutions is the support of single-sign-on across multiple service providers; the navigation from one service provider to another must be transparent for the

¹ Most prominently, the Austrian Citizen Cards refrain from using X.509 certificates for authentication in order to implement their privacy-enhancement approach.

user without requiring a repeated authentication.

To understand this requirement, it is crucial to distinguish between the user's perception of the process and the underlying authentication protocol. In a username/password approach where authentication is implemented as a “login page”, this distinction is not evident; the verification of the credential by the authentication protocol and the expression of user-consent to use the credential are the same, i.e., the user entering a password.

In other approaches that distinguish authentication with the credential and user-consent, user-perception and what happens underneath are independent. For example, an X.509 token can potentially sign many challenge and responses after a single expression of user-consent, i.e., entering the PIN or token password. Unless the middleware of the X.509 token enforces a different regime, the user-consent can be valid until the token is physically removed from the reader.

This explains why username/password approaches require “additional plumbing” to achieve single-sign-on from a user-experience point of view, while other credential types such as X.509 tokens can provide this out of the box.

Both, Liberty Alliance and WS-* use a “plumbing” approach to single-sign-on where the Identity Provider keeps a session with the user while for every change of service provider, a new session credential has to be issued. TLS-Federation in contrast uses the intrinsic single-sign-on capability of X.509 tokens.

The reason why this is only possible in TLS-Federation lies in the way session credentials are validated. In Liberty Alliance and WS-*, anyone who presents a session credential to a relying party is accepted as legitimate owner. These solutions thus use a large number of counter measures that prevent the “stealing” of credentials; a key counter measure being the drastic limitation of the time in which the session credential is valid².

TLS (and thus TLS-Federation) takes a completely different approach to the validation of session credentials. The decision on the validity of a credential is based on a challenge-and-response and the counter measures lie in the protection of the private key. The latter can be done by a secure storage device such as a smart card for static key pairs or by the short-livedness and possibly single-use of a key pair generated dynamically by the middleware. In both cases, stealing a session credential is considerably more difficult than just intercepting the session credential as it travels from Identity Provider to the user and then to the relying party.

Evidently, the necessary short-livedness of Liberty and WS-* session credentials is in direct contradiction to a plumbing-less single-sign-on feature. TLS-Federation, through the use of challenge-and-response avoids this limitation.

Authentication in Transport Layer Security

The optional authentication of the client during the TLS-handshake is specified in RFC 2246 “The TLS Protocol, Version 1.0” [xxxx], an Internet Engineering Task Force (IETF) standard that has been stable since 1999 and is ubiquitously implemented in a very wide range of software, including all major browsers.

The detailed description of the handshake is left to the standard document and accompanying literature (e.g., [XXX Netscape or Apache paper..]); the discussion of this section concentrates solely on the most important aspects and consequences.

It is important for judging the security of TLS-authentication to note that the handshake not only performs a one time authentication, but also results in an authenticated session. In TLS, a session

² In Liberty Alliance, this is done by using the NotValidBefore and NotValidAfter elements together with the avoidance of clock differences between Identity Provider and Relying Party

is created at transport layer, well before application packages such as HTTP are exchanged. The extensive security analysis that is available for TLS covers the integrally linked concepts of authentication and session.

This contrasts with the authentication concept of other approaches such as Liberty Alliance. Here, authentication is disconnected from the session concept and sessions have to be provided by other means. Typically, these authentication systems use cookies at the HTTP layer to implement session. While Liberty Alliance standardizes certain aspects of authentication, it fails to go in the merit of session management. Evidently, a complete security analysis has to cover both authentication and session management, since taking over sessions is likely a more relevant vector of attack than authentication. A recent discussion on a security list of common vulnerabilities of corporate single sign on solutions looks at these issues and proposes TLS (and a reverse proxy architecture) as a possible solution [xxx find links xxxx].

Browser Middleware for TLS-Authentication

It is useful to discuss the authentication process in the context of a web browser that uses middleware to access a credential, e.g., an eID. While the two major browsers use different APIs for this middleware, it is well standardized in both cases and most credential issuers package middleware for both browsers with their offerings. In particular, Internet Explorer uses the Cryptographic Service Provider (CSP) API [xxxx], while the Mozilla family of browsers uses the RSA standard PKCS#11 [xxxx]. As bridge software [xxx] that translates from one API to the other illustrate, for our purposes the APIs are conceptually equivalent.

For authentication purposes, the middleware provides access to three key functions:

- it asks user consensus for the access to the credential; typically this is done by entering a PIN or a pass-phrase
- it extracts an X.509 certificate from the credential and hands it to the browser for use in the TLS-handshake
- it interfaces with the credential to sign a challenge with the private key that corresponds to the public key contained in the certificate.

It is important to note that these local APIs fail to impose any restrictions on where the credential is actually located. The most common case is a local X.509 token such as a smart card, but any remote service capable of providing the necessary functionality is equally covered by the mentioned standards.

Federation in Transport Layer Security

TLS-Federation is a simple extension to plain TLS that defines how to interact with remote credentials. In other words, the Federation extension is not necessary for local X.509 credentials but is only necessary for other kinds of credentials that are incapable of autonomously supporting a TLS handshake but can do so only in collaboration with a remote Identity Provider service. In some cases, a remote Identity Provider can also be used with local X.509 credentials, for example to enhance its privacy characteristics.

The federation extension seamlessly integrates with plain TLS since the distinction between the federated or the plain version is hidden behind the API of the middleware and browsers can treat the federated and local case exactly the same.

The essence of TLS-Federation is a concrete choice of an existing standard to govern the interaction between the middleware and the remote Identity Provider. This interaction covers the following main tasks:

- the middleware authenticate to the Identity Provider
- it request an X.509 certificate for a pre-existing key pair

This is exactly the interaction used between a Registration Authority (RA) and a Certification Authority (CA) in a standard PKI scenario. Not surprisingly, this functionality is already well-standardized by the Internet Engineering Task Force. In particular, RFC 4210 “Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)” [xx] covers the necessary functionality.

The browser middleware thus assumes the role of an RA; the remote Identity Provider is nothing but a standard CA. Since this functionality is standard for any PKI setup, it is ubiquitously available in off-the-shelf software, including several open source implementations.

In a federated scenario, TLS authentication performs the following steps:

- the browser requests a certificate from the middleware through the standard API (CSP or PKCS#11)
- the middleware creates a key pair or decides to use a pre-existing one
- it authenticates to the Identity Provider and
- sends a standard certification request
- the middleware passes the resulting certificate to the browser
- browser asks middleware to sign the challenge it received from the browser with private key that corresponds to the public key of the certificate.

The concepts of federation are satisfied as illustrated in the following:

Federation of Trust:

All certificates used in TLS-Federation are issued by a Certification Authority; either a “static” one that issues a local X.509 credential of a long validity period, or a dynamic one (i.e., an Identity Provider) that issues X.509 certificates possibly of short validity period on the fly.

Federation of trust thus follows the standard scenario of a multi-CA PKI setup.

Independence of Credential Technology:

Certification Requests according to RFC 4210 foresee an arbitrary pluggable authentication mechanism. ????????????

[Detlef: how is authentication of the RA to the CA dealt with in RFC4210? It seems that PKCS#10 is possible but discouraged... Is this really general and not restrictive? Is there something we need to check in more detail???)

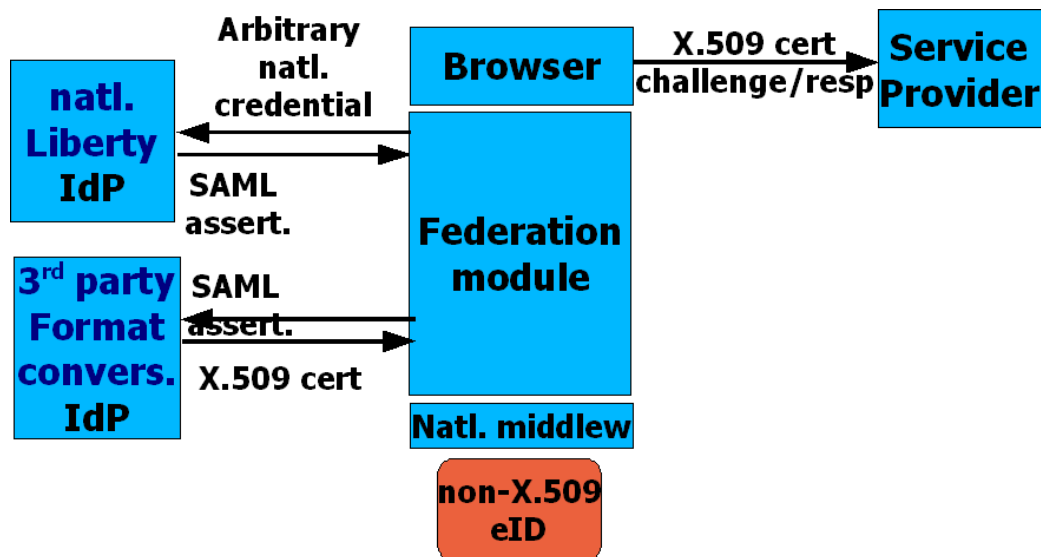
The independence of credential technology originates in the setup where the credential (e.g., a national eID) is used to authenticate to the Identity Provider (i.e., CA), not the relying part directly. The Identity Provider is free to support any kind (and number) of authentication protocols/credential types. Handling the certification request is nothing but verifying the authentication and translating the received credential into an X.509 credential that can be presented to the relying party.

Single Sign On:

Single sign on is implemented by the middleware in a way that requires the user to express consensus to use the credential only once even if it is then used to sign multiple challenges. In other words, while the underlying protocol requires a new authentication handshake at every change of relying partner/service provider, the middleware renders this transparent to the user.

More concretely, in PKCS#11 speak, the middleware logs into the token only once and refrains from logging out until single sign on is no longer required. The logout can in many cases occur when physically disconnecting the token from the host PC (e.g., by removing a smartcard from the reader or unplugging a USB token).

Working with Multiple Identity Providers



see my Porvoo 11 presentation for an example..: more than one session credential technology.. very flexible, open to third parties to get things done..

Multiple IdPs (see Porvoo 11 slides..)

only possible with user-centric approach. Doesn't apply to Liberty..

Also, migration scenarios from one to another technology??? Maybe multiple IdPs in a country, ...

Privacy Issues

cert with national unique id hard coded in auth certificate,

IdP derives Austrian style sector specific identifier (middleware must specify sector -> requires Identity Agent),

middleware creates new keypair at every session >> not linkable..

Conclusions

References