ABFAB
Internet-Draft
Intended status: Informational
Expires: August 29, 2013

J. Howlett
JANET(UK)
S. Hartman
Painless Security
H. Tschofenig
Nokia Siemens Networks
E. Lear
Cisco Systems GmbH
J. Schaad
Soaring Hawk Consulting

February 25, 2013

Application Bridging for Federated Access Beyond Web (ABFAB) Architecture
draft-ietf-abfab-arch-05.txt

Abstract

Over the last decade a substantial amount of work has occurred in the space of federated access management. Most of this effort has focused on two use-cases: network access and web-based access. However, the solutions to these use-cases that have been proposed and deployed tend to have few common building blocks in common.

This memo describes an architecture that makes use of extensions to the commonly used security mechanisms for both federated and non-federated access management, including the Remote Authentication Dial In User Service (RADIUS) and the Diameter protocol, the Generic Security Service (GSS), the GS2 family, the Extensible Authentication Protocol (EAP) and the Security Assertion Markup Language (SAML). The architecture addresses the problem of federated access management to primarily non-web-based services, in a manner that will scale to large numbers of identity providers, relying parties, and federations.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 29, 2013.

Copyright Notice

Table of Contents

1. Introduction 4
1.1. Terminology 5
1.1.1. Channel Binding 6
1.2. An Overview of Federation 7
1.3. Challenges for Contemporary Federation 10
1.4. An Overview of ABFAB-based Federation 11
1.5. Design Goals 13
2. Architecture 15
2.1. Relying Party to Identity Provider 16
2.1.1. AAA, RADIUS and Diameter 17
2.1.2. Discovery and Rules Determination 19
2.1.3. Routing and Technical Trust 20
2.1.4. AAA Security 21
2.1.5. SAML Assertions 22
2.2. Client To Identity Provider 24
2.2.1. Extensible Authentication Protocol (EAP) 24
2.2.2. EAP Channel Binding 25
2.3. Client to Relying Party 26
2.3.1. GSS-API 26
2.3.2. Protocol Transport 28
2.3.3. Reauthentication 28
3. Application Security Services 29
3.1. Authentication 29
3.2. GSS-API Channel Binding 30
3.3. Host-Based Service Names 31
3.4. Additional GSS-API Services 33
4. Privacy Considerations 34
4.1. Entities and their roles 34
4.2. Relationship between user and entities 35
4.3. Data and Identifiers in use 35
4.3.1. NAI 35
4.3.2. Identity Information 36
4.3.3. Accounting Information 36
4.3.4. Collection and retention of data and identifiers . . . 36
4.4. User Participation 37
5. Deployment Considerations 38
5.1. EAP Channel Binding 38
5.2. AAA Proxy Behavior 38
6. Security Considerations 39
7. IANA Considerations 41
8. Acknowledgments 42
9. References 43
9.1. Normative References 43
9.2. Informative References 43
Editorial Comments . . . . .
Authors' Addresses 49

1. Introduction

The Internet uses numerous security mechanisms to manage access to various resources. These mechanisms have been generalized and scaled over the last decade through mechanisms such as Simple Authentication and Security Layer (SASL) with the Generic Security Server Application Program Interface (GSS-API) (known as the GS2 family) [RFC5801], Security Assertion Markup Language (SAML) [OASIS.saml-core-2.0-os], and the Authentication, Authorization, and Accounting (AAA) architecture as embodied in RADIUS [RFC2865] and Diameter [RFC3588].

A Relying Party (RP) is the entity that manages access to some resource. The actor that is requesting access to that resource is often described as the Client. Many security mechanisms are manifested as an exchange of information between these actors. The RP is therefore able to decide whether the Client is authorized, or not.

Some security mechanisms allow the RP to delegate aspects of the access management decision to an actor called the Identity Provider (IdP). This delegation requires technical signaling, trust and a common understanding of semantics between the RP and IdP. These aspects are generally managed within a relationship known as a 'federation'. This style of access management is accordingly described as 'federated access management'.

Federated access management has evolved over the last decade through specifications like SAML [OASIS.saml-core-2.0-os], OpenID [1], OAuth [RFC5849], [I-D.ietf-oauth-v2] and WS-Trust [WS-TRUST]. The benefits of federated access management include:

Single or Simplified sign-on:

An Internet service can delegate access management, and the associated responsibilities such as identity management and credentialing, to an organisation that already has a long-term relationship with the Subject. This is often attractive for Relying Parties who frequently do not want these responsibilities. The Subject also requires fewer credentials, which is also desirable.

Data Minimization and User Participation:

Often a Relying Party does not need to know the identity of a Subject to reach an access management decision. It is frequently only necessary for the Relying Party know specific attributes about the subject, for example, that the Subject is affiliated

with a particular organisation or has a certain role or entitlement. Sometimes the RP only needs to know a pseudonym of the Subject.

Prior to the release of attributes to the IdP from the IdP, the IdP will check configuration and policy to determine if the attributes are to be released. There is currently no direct client participation in this decision.

Provisioning

Sometimes a Relying Party needs, or would like, to know more about a subject than an affiliation or a pseudonym. For example, a Relying Party may want the Subject's email address or name. Some federated access management technologies provide the ability for the IdP to supply this information, either on request by the RP or unsolicited.

This memo describes the Application Bridging for Federated Access Beyond the Web (ABFAB) architecture. This architecture makes use of extensions to the commonly used security mechanisms for both federated and non-federated access management, including the RADIUS and the Diameter protocols, the Generic Security Service (GSS), the GS2 family, the Extensible Authentication Protocol (EAP) and SAML. The architecture addresses the problem of federated access management primarily for non-web-based services. It does so in a manner that will scale to large numbers of identity providers, relying parties, and federations.

1.1. Terminology

This document uses identity management and privacy terminology from [I-D.iab-privacy-considerations]. In particular, this document uses the terms identity provider, relying party, identifier, pseudonymity, unlinkability, and anonymity.

In this architecture the IdP consists of the following components: an EAP server, a RADIUS or a Diameter server, and optionally a SAML Assertion service.

This document uses the term Network Access Identifier (NAI), as defined in [RFC4282]. An NAI consists of a realm identifier, which is associated with an IdP and a username which is associated with a specific client of the IdP.

One of the problems people will find with reading this document is that the terminology sometimes appears to be inconsistent. This is due the fact that the terms used by the different standards we are

picking up don't use the same terms. In general the document uses either a consistent term or the term associated with the standard under discussion as appropriate. For reference we include this table which maps the different terms into a single table.

Protocol	Subject	Relying Party	Identity Provider
ABFAB	Client	Relying Party (RP)	Identity Provider (IdP)
	Initiator	Acceptor	
		Server	
SAML	Subject	Service Provider	Issuer
GSS-API	Initiator	Acceptor	
EAP	EAP peer		EAP server
AAA		AAA Client	AAA server
RADIUS	user	NAS	RADIUS server
		RADIUS client	

Note that in some cases a cell has been left empty, in these cases there is no direct name that represents this concept.

Note to reviewers - I have most likely missed some entries in the table. Please provide me with both correct names from the protocol and missing names that are used in the text below.

1.1.1. Channel Binding

This document uses the term channel binding with two different meanings.

EAP channel binding, also called channel binding, is used to provide GSS-API naming semantics. Channel binding sends a set of attributes from the peer to the EAP server either as part of the EAP converstaion or as part of a secure association protocol. In addition, attributes are sent in the baackend protocol from the authenticator to the EAP server. The EAP server confirms the consistency of these attributes and provides the confirmation back to the peer.

GSS-API channel binding provides protection against man-in-the-middle attacks when GSS-API is used for authentication inside of some tunnel; it is similar to a facility called cryptographic binding in EAP. The binding works by each side deriving a cryptographic value from the tunnel itself and then using that cyrptographic value to prove to the otherside that it knows the value.

See [RFC5056] for a discussion of the differences between these two facilities.

Typically when considering channel binding, people think of channel binding in combination with mutual authentication. This is sufficiently common that without additional qualification channel binding should be assumed to imply mutual authentication. Without mutual authentication, only one party knows that the endpoints are correct. That's sometimes useful. Consider for example a user who wishes to access a protected resource from a shared whiteboard in a conference room. The whiteboard is the initiator; it does not need to actually authenticate that it is talking to the correct resource because the user will be able to recognize whether the displayed content is correct. If channel binding were used without mutual authentication, it would in effect be a request to only disclose the resource in the context of a particular channel. Such an authentication would be similar in concept to a holder-of-key SAML assertion. However, also note that while it is not happening in the protocol, mutual authentication is happening in the overall system: the user is able to visually authenticate the content. This is consistent with all uses of channel binding without protocol level mutual authentication found so far.

1.2. An Overview of Federation

In the previous section we introduced the following actors:

- the Client,
- the Identity Provider, and
- the Relying Party.

One additional actor in can be an Individual. An individual is a human being that is using a client. Individuals may or may not exist in any given deployment. The client may be either a front end on an individual or an independent automated entity.

These entities and their relationships are illustrated graphically in Figure 1.

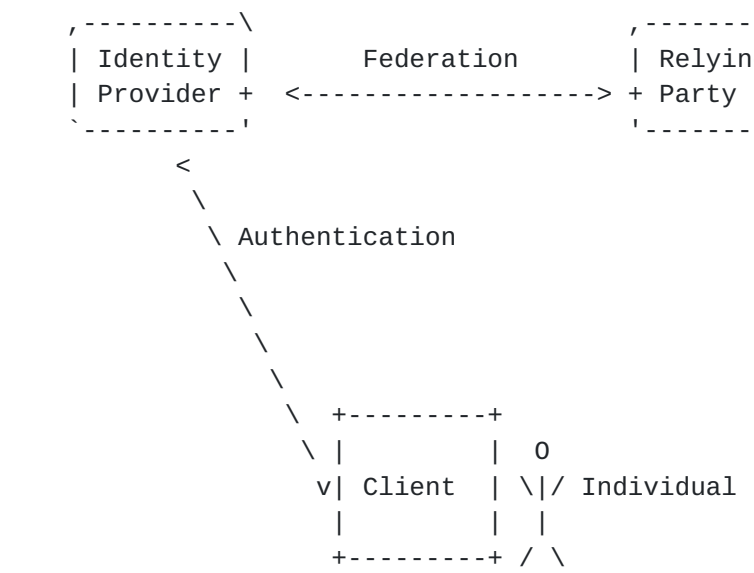


Figure 1: Entities and their Relationships

The relationships between the entities in Figure 1 are:

Federation

The Identity Provider and the Relying Parties are part of a Federation. The relationship may be direct (they have an explicit trust relationship) or transitive (the trust releationship is mediated by one or more entities). The federation relationship is governed by a federation agreement. Within a single federation, there may be multiple Identity Providers as well as multiple Relying Parties. A federation is governed by a federation agreement.

Authentication

There is a direct relationship between the Client and the Identity Provider by which the entities trust and can securely authenticate each other.

A federation agreement typically encompasses operational specifications and legal rules:

Operational Specifications:

These includes the technical specifications (e.g. protocols used to communicate between the three parties), process standards, policies, identity proofing, credential and authentication algorithm requirements, performance requirements, assessment and audit criteria, etc. The goal of operational specifications is to

provide enough definition that the system works and interoperability is possible.

Legal Rules:

The legal rules take the legal framework into consideration and provides contractual obligations for each entity. The rules define the responsibilities of each party and provide further clarification of the operational specifications. These legal rules regulate the operational specifications, make operational specifications legally binding to the participants, define and govern the rights and responsibilities of the participants. The legal rules may, for example, describe liability for losses, termination rights, enforcement mechanisms, measures of damage, dispute resolution, warranties, etc.

The Operational Specifications can demand the usage of a sophisticated technical infrastructure, including requirements on the message routing intermediaries, to offer the required technical functionality. In other environments, the Operational Specifications require fewer technical components in order to meet the required technical functionality.

The Legal Rules include many non-technical aspects of federation, such as business practices and legal arrangements, which are outside the scope of the IETF. The Legal Rules can still have an impact the architectural setup or on how to ensure the dynamic establishment of trust.

While a federation agreement is often discussed within the context of formal relationships, such as between an enterprise and an employee or a government and a citizen, a federation agreement does not have to require any particular level of formality. For an IdP and a Client, it is sufficient for a relationship to be established by something as simple as using a web form and confirmation email. For an IdP and an RP, it is sufficient for the IdP to publish contact information along with a public key and for the RP to use that data. With in the framework of ABFAB, it will generally be required that a mechanism exists for the IdP to be able to trust the identity of the RP, if this is not present then the IdP cannot provide the assurances to the client that the identity of the RP has been established.

The nature of federation dictates that there is some form of relationship between the identity provider and the relying party. This is particularly important when the relying party wants to use information obtained from the identity provider for access management decisions and when the identity provider does not want to release information to every relying party (or only under certain

conditions).

While it is possible to have a bilateral agreement between every IdP and every RP; on an Internet scale this setup requires the introduction of the multi-lateral federation concept, as the management of such pair-wise relationships would otherwise prove burdensome.

The IdP will typically have a long-term relationship with the Client. This relationship typically involves the IdP positively identifying and credentialing the Client (for example, at time of employment within an organization). When dealing with individuals, this process is called identity proofing [NIST-SP.800-63]. The relationship will often be instantiated within an agreement between the IdP and the Client (for example, within an employment contract or terms of use that stipulates the appropriate use of credentials and so forth).

The nature and quality of the relationship between the Subject and the IdP is an important contributor to the level of trust that an RP may attribute to an assertion describing a Client made by an IdP. This is sometimes described as the Level of Assurance [NIST-SP.800-63].

Federation does not require an a priori relationship or a long-term relationship between the RP and the Client; it is this property of federation that yields many of its benefits. However, federation does not preclude the possibility of a pre-existing relationship between the RP and the Client, nor that they may use the introduction to create a new long-term relationship independent of the federation.

Finally, it is important to reiterate that in some scenarios there might indeed be an Individual behind the Client and in other cases the Client may be autonomous.

1.3. Challenges for Contemporary Federation

As the number of federated services has proliferated, the role of the individual can become ambiguous in certain circumstances. For example, a school might provide online access for a student's grades to their parents for review, and to the student's teacher for modification. A teacher who is also a parent must clearly distinguish her role upon access.

Similarly, as the number of federations proliferates, it becomes increasingly difficult to discover which identity provider(s) a user is associated with. This is true for both the web and non-web case, but is particularly acute for the latter as many non-web authentication systems are not semantically rich enough on their own

to allow for such ambiguities. For instance, in the case of an email provider, the use of SMTP and IMAP protocols do not have the ability for the server to get additional information, beyond the clients NAI, in order to provide additional input to decide between multiple federations it may be associated with. However, the building blocks do exist to add this functionality.

1.4. An Overview of ABFAB-based Federation

The previous section described the general model of federation, and its the application of federated access management. This section provides a brief overview of ABFAB in the context of this model.

In this example, a client is attempting to connect to a server in order to either get access to some data or perform some type of transaction. In order for the client to mutually authenticate with the server, the following steps are taken in an ABFAB federated architecture:

1. Client Configuration: The Client Application is configured with an NAI assigned by the IdP. It is also configured with any keys, certificates, passwords or other secret and public information needed to run the EAP protocols between it and the IdP.

2. Authentication mechanism selection: The GSS-EAP GSS-API mechanism is selected for authentication/authorization.

3. Client provides an NAI to RP: The client application sets up a transport to the RP and begins the GSS-EAP authentication. In response, the RP sends an EAP request message (nested in the GSS-EAP protocol) asking for the Client's name. The Client sends an EAP response with an NAI name form that at a minimum, contains the realm portion of it's full NAI.

4. Discovery of federated IdP: The RP uses pre-configured information or a federation proxy to determine what IdP to use based on policy and the realm portion of the provided Client NAI. This is discussed in detail below (Section 2.1.2).

5. Request from Relying Party to IdP: Once the RP knows who the IdP is, it (or its agent) will send a RADIUS/Diameter request to the IdP. The RADIUS/Diameter access request encapsulates the EAP response. At this stage, the RP will likely have no idea who the client is. The RP sends its identity to the IdP in AAA attributes, and it may send a SAML Attribute Requests in a AAA attribute. The AAA network checks that the identity claimed by the RP is valid.

6. IdP begins EAP with the client: The IdP sends an EAP message to the client with an EAP method to be run. The IdP may re-request the clients name in this message, but this is unexpected behavior. The available and appropriate methods are discussed below in this memo (Section 2.2.1).

7. The EAP protocol is run: A bunch of EAP messages are passed between the client (EAP peer) and the IdP (EAP server), until the result of the authentication protocol is determined. The number and content of those messages depends on the EAP method selected. If the IdP is unable to authenticate the client, the IdP sends a EAP failure message to the RP. As part of the EAP protocol, the client sends a channel bindings EAP message to the IdP (Section 2.2.2). In the channel binding message the client identifies, among other things, the RP to which it is attempting to authenticate. The IdP checks the channel binding data from the client with that provided by the RP via the AAA protocol. If the bindings do not match the IdP sends an EAP failure message to the RP.

8. Successful EAP Authentication: At this point, the IdP (EAP server) and client (EAP peer) have mutually authenticated each other. As a result, the subject and the IdP hold two cryptographic keys: a Master Session Key (MSK), and an Extended MSK (EMSK). At this point the client has a level of assurance about the identity of the RP based on the name checking the IdP has done using the RP naming information from the AAA framework and from the client (by the channel binding data).

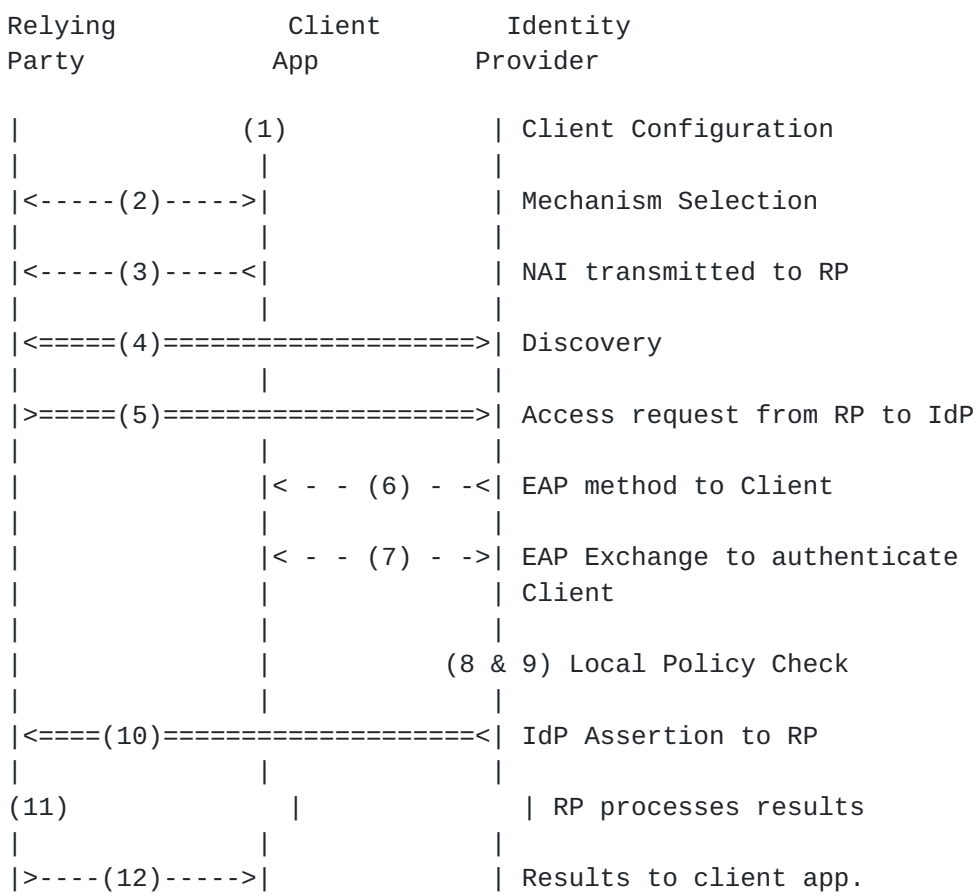
9. Local IdP Policy Check: At this stage, the IdP checks local policy to determine whether the RP and client are authorized for a given transaction/service, and if so, what if any, attributes will be released to the RP. If the IdP gets a policy failure, it sends an EAP failure message to the RP.[anchor4] (The RP will have done its policy checks during the discovery process.)

10. IdP provide the RP with the MSK: The IdP sends a positive result EAP to the RP, along with an optional set of AAA attributes associated with the client (usually as one or more SAML assertions). In addition, the EAP MSK is returned to the RP.

11. RP Processes Results: When the RP receives the result from the IdP, it should have enough information to either grant or refuse a resource access request. It may have information that associates the client with specific authorization identities. If additional attributes are needed from the IdP the RP may make a new SAML Request to the IdP. It will apply these results in an application-specific way.

12. RP returns results to client: Once the RP has a response it must inform the client application of the result. If all has gone well, all are authenticated, and the application proceeds with appropriate authorization levels. The client can now complete the authentication of the RP by the use of the EAP MSK value.

An example communication flow is given below:



----- = Between Client App and RP
===== = Between RP and IdP
- - - = Between Client App and IdP

1.5. Design Goals

Our key design goals are as follows:

- Each party of a transaction will be authenticated, although perhaps not identified, and the client will be authorized for access to a specific resource.

- Means of authentication is decoupled so as to allow for multiple authentication methods.

- Hence, the architecture requires no sharing of long term private keys between clients and servers.

- The system will scale to large numbers of identity providers, relying parties, and users.

- The system will be designed primarily for non-Web-based authentication.

- The system will build upon existing standards, components, and operational practices.

Designing new three party authentication and authorization protocols is hard and fraught with risk of cryptographic flaws. Achieving widespead deployment is even more difficult. A lot of attention on federated access has been devoted to the Web. This document instead focuses on a non-Web-based environment and focuses on those protocols where HTTP is not used. Despite the increased excitement for layering every protocol on top of HTTP there are still a number of protocols available that do not use HTTP-based transports. Many of these protocols are lacking a native authentication and authorization framework of the style shown in Figure 1.

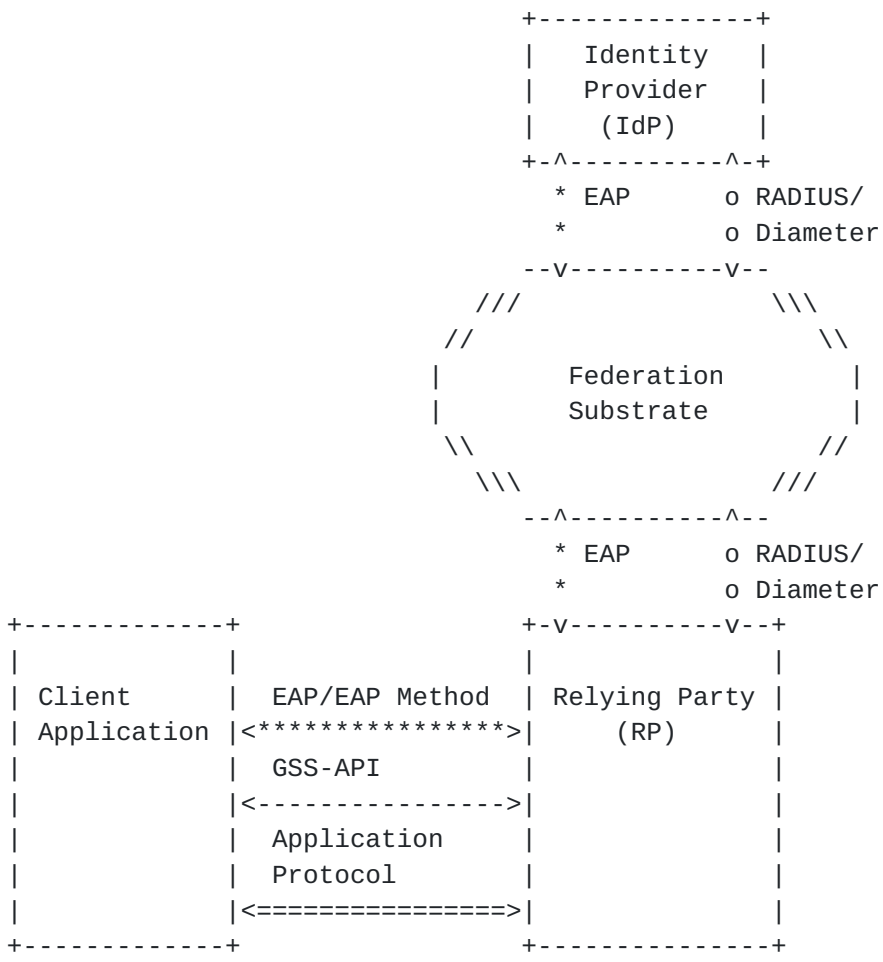
2. Architecture

We have already introduced the federated access architecture, with the illustration of the different actors that need to interact, but did not expand on the specifics of providing support for non-Web based applications. This section details this aspect and motivates design decisions. The main theme of the work described in this document is focused on re-using existing building blocks that have been deployed already and to re-arrange them in a novel way.

Although this architecture assumes updates to the relying party, the client application, and the Identity Provider, those changes are kept at a minimum. A mechanism that can demonstrate deployment benefits (based on ease of update of existing software, low implementation effort, etc.) is preferred and there may be a need to specify multiple mechanisms to support the range of different deployment scenarios.

There are a number of ways for encapsulating EAP into an application protocol. For ease of integration with a wide range of non-Web based application protocols the usage of the GSS-API was chosen. A description of the technical specification can be found in [I-D.ietf-abfab-gss-eap]. Other alternatives exist as well and may be considered later, such as "TLS using EAP Authentication" [I-D.nir-tls-eap]. [anchor7]

The architecture consists of several building blocks, which is shown graphically in Figure 2. In the following sections, we discuss the data flow between each of the entities, the protocols used for that data flow and some of the trade-offs made in choosing the protocols.



Legend:

<****>: Client-to-IdP Exchange
<---->: Client-to-RP Exchange
<oooo>: RP-to-IdP Exchange
<====>: Protocol through which GSS-API/GS2 exchanges are tunneled

Figure 2: ABFAB Protocol Instantiation

2.1. Relying Party to Identity Provider

Communications between the Relying Party and the Identity Provider is done by the federation substrate. This communication channel is responsible for:

o Establishing the trust relationship between the RP and the IdP.

o Determining the rules governing the relationship.

- Conveying authentication packets from the client to the IdP and back.

- Providing the means of establishing a trust relationship between the RP and the client.

- Providing a means for the RP to obtain attributes about the client from the IdP.

The ABFAB working group has chosen the AAA framework for the messages transported between the RP and IdP. The AAA framework supports the requirements stated above as follows:

- The AAA backbone supplies the trust relationship between the RP and the IdP.

- The agreements governing a specific AAA backbone contains the rules governing the relationships within the AAA federation.

- A method exists for carrying EAP packets within RADIUS [RFC3579] and Diameter [RFC4072].

- The use of EAP channel binding [RFC6677] along with the core ABFAB protocol provide the pieces necessary to establish the identities of the RP and the client, while EAP provides the cryptographic methods for the RP and the client to validate they are talking to each other.

- A method exists for carrying SAML packets within RADIUS [I-D.ietf-abfab-aaa-saml] and Diameter (work in progress) which allows the RP to query attributes about the client from the IdP.

Future protocols that support the same framework but do different routing may be used in the future. Once such effort is to setup a framework that creates a trusted point-to-point channel on the fly.

2.1.1. AAA, RADIUS and Diameter

Interestingly, for network access authentication the usage of the AAA framework with RADIUS [RFC2865] and Diameter [RFC3588] was quite successful from a deployment point of view. To map the terminology used in Figure 1 to the AAA framework the IdP corresponds to the AAA server, the RP corresponds to the AAA client, and the technical building blocks of a federation are AAA proxies, relays and redirect agents (particularly if they are operated by third parties, such as AAA brokers and clearing houses). The front-end, i.e. the end host to AAA client communication, is in case of network access authentication offered by link layer protocols that forward

authentication protocol exchanges back-and-forth. An example of a large scale RADIUS-based federation is EDUROAM [2].

By using the AAA framework, ABFAB gets a lot of mileage as many of the federation agreements already exist and merely need to be expanded to cover the ABFAB additions. The AAA framework has already addressed some of the problems outlined above. For example,

- It already has a method for routing requests based on a domain.
- It already has an extensible architecture allowing for new attributes to be defined and transported.
- Pre-existing relationships can be re-used.

The astute reader will notice that RADIUS and Diameter have substantially similar characteristics. Why not pick one? RADIUS and Diameter are deployed in different environments. RADIUS can often be found in enterprise and university networks, and is also in use by fixed network operators. Diameter, on the other hand, is deployed by mobile operators. Another key difference is that today RADIUS is largely transported upon UDP. We leave as a deployment decision, which protocol will be appropriate. The protocol defines all the necessary new AAA attributes as RADIUS attributes. A future document would defined the same AAA attributes for a Diameter environment. We also note that there exist proxies which convert from RADIUS to Diameter and back. This makes it possible for both to be deployed in a single federation substrate.

Through the integrity protection mechanisms in the AAA framework, the identity provider can establish technical trust that messages are being sent by the appropriate relying party. Any given interaction will be associated with one federation at the policy level. The legal or business relationship defines what statements the identity provider is trusted to make and how these statements are interpreted by the relying party. The AAA framework also permits the relying party or elements between the relying party and identity provider to make statements about the relying party.

The AAA framework provides transport for attributes. Statements made about the subject by the identity provider, statements made about the relying party and other information are transported as attributes.

One demand that the AAA substrate makes of the upper layers is that they must properly identify the end points of the communication. It must be possible for the AAA client at the RP to determine where to send each RADIUS or Diameter message. Without this requirement, it would be the RP's responsibility to determine the identity of the

client on its own, without the assistance of an IdP. This architecture makes use of the Network Access Identifier (NAI), where the IdP is indicated by the realm component [RFC4282]. The NAI is represented and consumed by the GSS-API layer as GSS_C_NT_USER_NAME as specified in [RFC2743]. The GSS-API EAP mechanism includes the NAI in the EAP Response/Identity message.

2.1.2. Discovery and Rules Determination

While we are using the AAA protocols to communicate with the IdP, the RP may have multiple federation substrates to select from. The RP has a number of criteria that it will use in selecting which of the different federations to use:

- The federation selected must be able to communicate with the IdP.
- The federation selected must match the business rules and technical policies required for the RP security requirements.

The RP needs to discover which federation will be used to contact the IdP. The first selection criteria in discovery is going to be the name of the IdP to be contacted. The second selection criteria in discovery is going to be the set of business rules and technical policies governing the relationship; this is called rules determination. The RP also needs to establish technical trust in the communications with the IdP.

Rules determination covers a broad range of decisions about the exchange. One of these is whether the given RP is permitted to talk to the IdP using a given federation at all, so rules determination encompasses the basic authorization decision. Other factors are included, such as what policies govern release of information about the principal to the RP and what policies govern the RP's use of this information. While rules determination is ultimately a business function, it has significant impact on the technical exchanges. The protocols need to communicate the result of authorization. When multiple sets of rules are possible, the protocol must disambiguate which set of rules are in play. Some rules have technical enforcement mechanisms; for example in some federations intermediaries validate information that is being communicated within the federation.

At the time of writing no protocol mechanism has been specified to allow a AAA client to determine whether a AAA proxy will indeed be able to route AAA requests to a specific IdP. The AAA routing is impacted by business rules and technical policies that may be quite complex and atpresent time, the route selection is based on manual configuration.

2.1.3. Routing and Technical Trust

Several approaches to having messages routed through the federation substrate are possible. These routing methods can most easily be classified based on the mechanism for technical trust that is used. The choice of technical trust mechanism constrains how rules determination is implemented. Regardless of what deployment strategy is chosen, it is important that the technical trust mechanism be able to validate the names of both parties to the exchange. The trust mechanism must to ensure that the entity acting as IdP for a given NAI is permitted to be the IdP for that realm, and that any service name claimed by the RP is permitted to be claimed by that entity. Here are the categories of technical trust determination:

AAA Proxy:
: The simplest model is that an RP supports a request directly to an AAA proxy. The hop-by-hop integrity protection of the AAA fabric provides technical trust. An RP can submit a request directly to a federation. Alternatively, a federation disambiguation fabric can be used. Such a fabric takes information about what federations the RP is part of and what federations the IdP is part of and routes a message to the appropriate federation. The routing of messages across the fabric plus attributes added to requests and responses provides rules determination. For example, when a disambiguation fabric routes a message to a given federation, that federation's rules are chosen. Name validation is enforced as messages travel across the fabric. The entities near the RP confirm its identity and validate names it claims. The fabric routes the message towards the appropriate IdP, validating the IdP's name in the process. The routing can be statically configured. Alternatively a routing protocol could be developed to exchange reachability information about given IdPs and to apply policy across the AAA fabric. Such a routing protocol could flood naming constraints to the appropriate points in the fabric.

Trust Broker:
: Instead of routing messages through AAA proxies, some trust broker could establish keys between entities near the RP and entities near the IdP. The advantage of this approach is efficiency of message handling. Fewer entities are needed to be involved for each message. Security may be improved by sending individual messages over fewer hops. Rules determination involves decisions made by trust brokers about what keys to grant. Also, associated with each credential is context about rules and about other aspects of technical trust including names that may be claimed. A routing protocol similar to the one for AAA proxies is likely to be useful to trust brokers in flooding rules and naming

constraints.

Global Credential:
A global credential such as a public key and certificate in a public key infrastructure can be used to establish technical trust. A directory or distributed database such as the Domain Name System is used by the RP to discover the endpoint to contact for a given NAI. Either the database or certificates can provide a place to store information about rules determination and naming constraints. Provided that no intermediates are required (or appear to be required) and that the RP and IdP are sufficient to enforce and determine rules, rules determination is reasonably simple. However applying certain rules is likely to be quite complex. For example if multiple sets of rules are possible between an IdP and RP, confirming the correct set is used may be difficult. This is particularly true if intermediates are involved in making the decision. Also, to the extent that directory information needs to be trusted, rules determination may be more complex.

Real-world deployments are likely to be mixtures of these basic approaches. For example, it will be quite common for an RP to route traffic to a AAA proxy within an organization. That proxy could then use any of the three methods to get closer to the IdP. It is also likely that rather than being directly reachable, the IdP may have a proxy on the edge of its organization. Federations will likely provide a traditional AAA proxy interface even if they also provide another mechanism for increased efficiency or security.

2.1.4. AAA Security

For the AAA framework there are two different places where security needs to be examined. The first is the security that is in place for the links in the AAA backbone being used. The second is the nodes that the backbone consists of.

The default link security for RADIUS is showing it's age as it uses MD5 and a shared secret to both obfuscate passwords and to provide integrity on the RADIUS messages. In many environments this is considered to be insufficient, especially as not all attributes are obfuscated and can thus leak information to a passive eavesdropper. The use of RADIUS with TLS [RFC6614] and/or DTLS [I-D.ietf-radext-dtls] addresses these attacks. The same level of security is included in the base Diameter specifications.

TBD - Put in text - Not all nodes can be eliminated - proxy nodes may be required Trust router looks for a way to shorten the list of inner nodes. Reference DYNAMIC and say that it does or does not help and

why. Talk about Diameter in the same context - does it have the same set of issues or not?

2.1.5. SAML Assertions

For the traditional use of AAA frameworks, network access, the only requirement that was necessary to grant access was an affirmative response from the IdP. In the ABFAB world, the RP may need to get additional information about the client before granting access. ABFAB therefore has a requirement that it can transport an arbitrary set of attributes about the client from the IdP to the RP.

Security Assertions Markup Language (SAML) [OASIS.saml-core-2.0-os] was designed in order to carry an extensible set of attributes about a subject. Since SAML is extensible in the attribute space, ABFAB has no immediate needs to update the core SAML specifications for our work. It will be necessary to update IdPs that need to return SAML assertions to IdPs and for both the IdP and the RP to implement a new SAML profile designed to carry SAML assertions in AAA. The new profile can be found in RFCXXXX [I-D.ietf-abfab-aaa-saml]. As SAML statements will frequently be large, RADIUS servers and clients that deal with SAML statements will need to implement RFC XXXX [I-D.perez-radext-radius-fragmentation]

There are several issues that need to be highlighted:

- The security of SAML assertions.
- Namespaces and mapping of SAML attributes.
- Subject naming of entities.
- Making multiple queries about the subject(s).
- Level of Assurance for authentication.

SAML assertions have an optional signature that can be used to protect and provide origination of the assertion. These signatures are normally based on asymmetric key operations and require that the verifier be able to check not only the cryptographic operation, but also the binding of the originators name and the public key. In a federated environment it will not always be possible for the RP to validate the binding, for this reason the technical trust established in the federation is used as an alternate method of validating the origination and integrity of the SAML Assertion.

Attributes placed in SAML assertions can have different namespaces assigned to the same name. In many, but not all, cases the

federation agreements will determine what attributes can be used in a SAML statement. This means that the RP needs to map from the federation names, types and semantics into the ones that the policies of the RP are written in. In other cases the federation substrate may modify the SAML assertions in transit to do the necessary namespace, naming and semantic mappings as the assertion crosses the different boundaries in the federation. If the proxies are modifying the SAML Assertion, then will obviously remove any signatures on the SAML assertions as they would no longer validate. In this case the technical trust is the required mechanism for validating the integrity of the assertion. Finally, the attributes may still be in the namespace of the originating IdP. When this occurs the RP will need to get the required mapping operations from the federation agreements and do the appropriate mappings itself.

As of this writing, no one has defined a SAML name format that corresponds to the NAI structure defined by RFC 4282 [RFC4282]. This means that there is no method to directly place the same NAI used in RADIUS or Diameter as the subject name of a SAML assertion. It is a requirement on the EAP server that it validate that the subject of the SAML name, if any, is equivalent to the subject identified by the NAI used in the RADIUS or Diameter session.

RADIUS has the ability to deal with multiple SAML queries for those EAP Servers which follow RFC 5080 [RFC5080]. In this case a State attribute will always been returned with the Access-Accept. The EAP client can then send a new Access-Request with the State attribute and the new SAML request Multiple SAML queries can them be done by making a new Access-Request using the State attribute returned in the last Access-Accept to link together the different RADIUS sessions.

Some RPs need to ensure that specfic criteria are met during the authentication process. This need is met by using Levels of Assurance. The way a Level of Assurance is communicated to from the RP to the EAP server is by the use of a SAML Authentication Request using the Authentication Profile from RFC XXX [I-D.ietf-abfab-aaa-saml] When crossing boundaries between different federations, either the policy specfied will need to be shared between the two federations, the policy will need to be mapped by the proxy server on the boundary or the proxy server on the boundary will need to supply infomration the EAP server so that it can do the required mapping. If this mapping is not done, then the EAP server will not be able to enforce the desired Level of Assurance as it will not understand the policy requirements.

2.2. Client To Identity Provider

Looking at the communications between the client and the IdP, the following items need to be dealt with:

- The client and the IdP need to mutually authenticate each other.
- The client and the IdP need to mutually agree on the identity of the RP.

ABFAB selected EAP for the purposes of mutual authentication and assisted in creating some new EAP channel binding documents for dealing with determining the identity of the RP. A framework for the channel binding mechanism has been defined in RFC 6677 [RFC6677] that allows the IdP to check the identity of the RP provided by the AAA framework with that provided by the client.

2.2.1. Extensible Authentication Protocol (EAP)

Traditional web federation does not describe how a subject interacts with an identity provider for authentication. As a result, this communication is not standardized. There are several disadvantages to this approach. Since the communication is not standardized, it is difficult for machines to correctly enter their credentials with different authentications, where Individuals can correctly identify the entyr mechanism on the fly. The use of browsers for authentication restricts the deployment of more secure forms of authentication beyond plaintext username and password known by the server. In a number of cases the authentication interface may be presented before the subject has adequately validated they are talking to the intended server. By giving control of the authentication interface to a potential attacker, then the security of the system may be reduced and phishing opportunities introduced.

As a result, it is desirable to choose some standardized approach for communication between the subject's end-host and the identity provider. There are a number of requirements this approach must meet.

Experience has taught us one key security and scalability requirement: it is important that the relying party not get possession of the long-term secret of the client. Aside from a valuable secret being exposed, a synchronization problem can develop when the client changes keys with the IdP.

Since there is no single authentication mechanism that will be used everywhere there is another associated requirement: The authentication framework must allow for the flexible integration of

authentication mechanisms. For instance, some IdPs require hardware tokens while others use passwords. A service provider wants to provide support for both authentication methods, and other methods from IdPs not yet seen.

Fortunately, these requirements can be met by utilizing standardized and successfully deployed technology, namely by the Extensible Authentication Protocol (EAP) framework [RFC3748]. Figure 2 illustrates the integration graphically.

EAP is an end-to-end framework; it provides for two-way communication between a peer (i.e,service client or principal) through the authenticator (i.e., service provider) to the back-end (i.e., identity provider). Conveniently, this is precisely the communication path that is needed for federated identity. Although EAP support is already integrated in AAA systems (see [RFC3579] and [RFC4072]) several challenges remain:

- The first is how to carry EAP payloads from the end host to the relying party.

- Another is to verify statements the relying party has made to the subject, confirm these statements are consistent with statements made to the identity provider and confirm all the above are consistent with the federation and any federation-specific policy or configuration.

- Another challenge is choosing which identity provider to use for which service.

The EAP method used for ABFAB needs to meet the following requirements:

- It needs to provide mutual authentication of the client and IdP.

- It needs to support channel binding.

As of this writing, the only EAP method that meets these criteria is TEAP [I-D.ietf-emu-eap-tunnel-method] either alone (if client certificates are used) or with an inner EAP method that does mutual authentication.

2.2.2. EAP Channel Binding

EAP channel binding is easily confused with a facility in GSS-API also called channel binding. GSS-API channel binding provides protection against man-in-the-middle attacks when GSS-API is used as authentication inside some tunnel; it is similar to a facility called

cryptographic binding in EAP. See [RFC5056] for a discussion of the differences between these two facilities and Section 6.1 for how GSS-API channel binding is handled in this mechanism.

The client knows, in theory, the name of the RP that it attempted to connect to, however in the event that an attacker has intercepted the protocol, the client and the IdP need to be able to detect this situation. A general overview of the problem along with a recommended way to deal with the channel binding issues can be found in RFC 6677 [RFC6677].

Since that document was published, a number of possible attacks were found and methods to address these attacks have been outlined in [I-D.ietf-emu-crypto-bind].

2.3. Client to Relying Party

The final set of interactions between parties to consider are those between the client and the RP. In some ways this is the most complex set since at least part of it is outside the scope of the ABFAB work. The interactions between these parties include:

- Running the protocol that implements the service that is provided by the RP and desired by the client.
- Authenticating the client to the RP and the RP to the client.
- Providing the necessary security services to the service protocol that it needs beyond authentication.
- Deal with client re-authentication where desired.

2.3.1. GSS-API

One of the remaining layers is responsible for integration of federated authentication into the application. There are a number of approaches that applications have adopted for security. So, there may need to be multiple strategies for integration of federated authentication into applications. However, we have started with a strategy that provides integration to a large number of application protocols.

Many applications such as SSH [RFC4462], NFS [RFC2203], DNS [RFC3645] and several non-IETF applications support the Generic Security Services Application Programming Interface [RFC2743]. Many applications such as IMAP, SMTP, XMPP and LDAP support the Simple Authentication and Security Layer (SASL) [RFC4422] framework. These two approaches work together nicely: by creating a GSS-API mechanism,

SASL integration is also addressed. In effect, using a GSS-API mechanism with SASL simply requires placing some headers on the front of the mechanism and constraining certain GSS-API options.

GSS-API is specified in terms of an abstract set of operations which can be mapped into a programming language to form an API. When people are first introduced to GSS-API, they focus on it as an API. However, from the prospective of authentication for non-web applications, GSS-API should be thought of as a protocol not an API. It consists of some abstract operations such as the initial context exchange, which includes two sub-operations (gss_init_sec_context and gss_accept_sec_context). An application defines which abstract operations it is going to use and where messages produced by these operations fit into the application architecture. A GSS-API mechanism will define what actual protocol messages result from that abstract message for a given abstract operation. So, since this work is focusing on a particular GSS-API mechanism, we generally focus on protocol elements rather than the API view of GSS-API.

The API view has significant value. Since the abstract operations are well defined, the set of information that a mechanism gets from the application is well defined. Also, the set of assumptions the application is permitted to make is generally well defined. As a result, an application protocol that supports GSS-API or SASL is very likely to be usable with a new approach to authentication including this one with no required modifications. In some cases, support for a new authentication mechanism has been added using plugin interfaces to applications without the application being modified at all. Even when modifications are required, they can often be limited to supporting a new naming and authorization model. For example, this work focuses on privacy; an application that assumes it will always obtain an identifier for the principal will need to be modified to support anonymity, unlinkability or pseudonymity.

So, we use GSS-API and SASL because a number of the application protocols we wish to federate support these strategies for security integration. What does this mean from a protocol standpoint and how does this relate to other layers? This means we need to design a concrete GSS-API mechanism. We have chosen to use a GSS-API mechanism that encapsulates EAP authentication. So, GSS-API (and SASL) encapsulate EAP between the end-host and the service. The AAA framework encapsulates EAP between the relying party and the identity provider. The GSS-API mechanism includes rules about how principals and services are named as well as per-message security and other facilities required by the applications we wish to support.

2.3.2. Protocol Transport

The transport of data between the client and the relying party is not provided by GSS-API. GSS-API creates and consumes messages, but it does not provide the transport itself, instead the protocol using GSS-API needs to provide the transport. In many cases HTTP or HTTPS is used for this transport, but other transports are perfectly acceptable. The core GSS-API document [RFC2743] provides some details on what requirements exist.

In addition we highlight the following:

- The transport does not need to provide either privacy or integrity. After GSS-EAP has finished negotiation, GSS-API can be used to provide both services. If the negotiation process itself needs protection from eavesdroppers then the transport would need to provide the necessary services.

- The transport needs to provide reliable transport of the messages.

- The transport needs to ensure that tokens are delivered in order during the negotiation process.

- GSS-API messages need to be delivered atomically. If the transport breaks up a message it must also reassemble the message before delivery.

2.3.3. Reauthentication

TBD.

3. Application Security Services

One of the key goals is to integrate federated authentication into existing application protocols and where possible, existing implementations of these protocols. Another goal is to perform this integration while meeting the best security practices of the technologies used to perform the integration. This section describes security services and properties required by the EAP GSS-API mechanism in order to meet these goals. This information could be viewed as specific to that mechanism. However, other future application integration strategies are very likely to need similar services. So, it is likely that these services will be expanded across application integration strategies if new application integration strategies are adopted.

3.1. Authentication

GSS-API provides an optional security service called mutual authentication. This service means that in addition to the initiator providing (potentially anonymous or pseudonymous) identity to the acceptor, the acceptor confirms its identity to the initiator. Especially for the ABFAB context, this service is confusingly named. We still say that mutual authentication is provided when the identity of an acceptor is strongly authenticated to an anonymous initiator.

RFC 2743, unfortunately, does not explicitly talk about what mutual authentication means. Within this document we therefore define it as:

- If a target name is configured for the initiator, then the initiator trusts that the supplied target name describes the acceptor. This implies both that appropriate cryptographic exchanges took place for the initiator to make such a trust decision, and that after evaluating the results of these exchanges, the initiator's policy trusts that the target name is accurate.

- If no target name is configured for the initiator, then the initiator trusts that the acceptor name, supplied by the acceptor, correctly names the entity it is communicating with.

- Both the initiator and acceptor have the same key material for per-message keys and both parties have confirmed they actually have the key material. In EAP terms, there is a protected indication of success.

Mutual authentication is an important defense against certain aspects of phishing. Intuitively, clients would like to assume that if some

party asks for their credentials as part of authentication, successfully gaining access to the resource means that they are talking to the expected party. Without mutual authentication, the server could "grant access" regardless of what credentials are supplied. Mutual authentication better matches this user intuition.

It is important, therefore, that the GSS-EAP mechanism implement mutual authentication. That is, an initiator needs to be able to request mutual authentication. When mutual authentication is requested, only EAP methods capable of providing the necessary service can be used, and appropriate steps need to be taken to provide mutual authentication. While a broader set of EAP methods could be supported by not requiring mutual authentication, it was decided that the client needs to always have the ability to request it. In some cases the IdP and the RP will not support mutual authentication, however the client will always be able to detect this and make an appropriate security decision.

The AAA infrastructure MAY hide the initiator's identity from the GSS-API acceptor, providing anonymity between the initiator and the acceptor. At this time, whether the identity is disclosed is determined by EAP server policy rather than by an indication from the initiator. Also, initiators are unlikely to be able to determine whether anonymous communication will be provided. For this reason, initiators are unlikely to set the anonymous return flag from GSS_Init_Sec_context.

3.2. GSS-API Channel Binding

[RFC5056] defines a concept of channel binding to which is used prevent man-in-the-middle attacks. The channel binding works by taking a cryptographic value from the transport security and checks that both sides of the GSS-API conversation know this value. Transport Layer Security (TLS) is the most common transport security layer used for this purpose.

It needs to be stressed that RFC 5056 channel binding (also called GSS-API channel binding when GSS-API is involved) is not the same thing as EAP channel binding. GSS-API channel binding is used for detecting Man-In-The-Middle attacks. EAP channel binding is used for mututal authentication and acceptor naming checks. Details are discussed in the mechanisms specification [I-D.ietf-abfab-gss-eap]. A fuller discription of the differences between the factilities cn be found in RFC 5056 [RFC5056].

The use of TLS can provide both encryption and integrity on the channel. It is common to provide SASL and GSS-API with these other security services.

On of the benifits that the use of TLS provides, is that client has the ability to validate the name of the server. However this validation is predicated on on a couple of things. The TLS sessions needs to be using certificates and not be an anonymous session. The client and the TLS need to share a common trust point for the certificate used in validating the server. TLS provides its own server authentication. However there are a variety of situations where this authentication is not checked for policy or usability reasons. Even when it is checked, if the trust infrastructure behind the TLS authentication is different from the trust infrastructure behind the GSS-API mutual authentication then confirming the end-points using both trust infrastructures is likely to enhance security. If the endpoints of the GSS-API authentication are different than the endpoints of the lower layer, this is a strong indication of a problem such as a man-in-the-middle attack. Channel binding provides a facility to determine whether these endpoints are the same.

The GSS-EAP mechanism needs to support channel binding. When an application provides channel binding data, the mechanism needs to confirm this is the same on both sides consistent with the GSS-API specification.

3.3. Host-Based Service Names

IETF security mechanisms typically take a host name and perhaps a service, entered by a user, and make some trust decision about whether the remote party in the interaction is the intended party. This decision can be made by the use of certificates, pre-configured key information or a previous leap of trust. GSS-API has defined a relatively flexible name convention, however most of the IETF applications that use GSS-API (including SSH, NFS, IMAP, LDAP and XMPP) have chosen to use a more restricted naming convention based on the host name. The GSS-EAP mechanism needs to support host-based service names in order to work with existing IETF protocols.

The use of host-based service names leads to a challenging trust delegation problem. Who is allowed to decide whether a particular host name maps to a specific entity. Possible solutions to this problem have been looked at.

The public-key infrastructure (PKI) used by the web has chosen to have a number of trust anchors (root certificate authorities) each of which can map any host name to a public key.

A number of GSS-API mechanisms, such as Kerberos [RFC1964], have split the problem into two parts. A new concept called a realm is introduced, the realm is responsible for host mapping within that

realm. The mechanism then decides what realm is responsible for a given name. This is the approach adopted by ABFAB.

GSS-EAP defines a host naming convention that takes into account the host name, the realm, the service and the service parameters. An example of GSS-API service name is "xmpp/foo@example.com". This identifies the XMPP service on the host foo in the realm example.com. Any of the components, except for the service name may be omitted from a name. When omitted, then a local default would be used for that component of the name.

While there is no requirement that realm names map to Fully Qualified Domain Names (FQDN) within DNS, in practice this is normally true. Doing so allows for the realm portion of service names and the portion of NAIs to be the same. It also allows for the use of DNS in locating the host of a service while establishing the transport channel between the client and the relying party.

It is the responsibility of the application to determine the server that it is going to communicate with, GSS-API has the ability to help confirm that the server is the desired server but not to determine the name of the server to use. It is also the responsibility of the application to determine how much of the information identifying the service needs to be validated by the ABFAB system. The information that needs to be validated is used to build up the service name passed into the GSS-EAP mechanism. What information is to be validated will depend on both what information was provided by the client, and what information is considered significant. If the client only cares about getting a specific service, then the host and realm that provides the service does not need to be validated.

In many cases applications may retrieve information about providers of services from DNS. When Service Records (SRV) and Naming Authority Pointer (NAPTR) records are used to help find a host that provides a service, the security requirements on the referrals is going to interact with the information used in the service name. If the a host name is returned from the DNS referrals, and the host name is to be validated by GS-EAP, then it makes sense that the referrals themselves should be secure. On the other hand, if the host name returned is not validated, i.e. only the service is passed in, then it is less important that the host name be obtained in a secure manner.

Another issue that needs to be addressed for host-based service names is that they do not work ideally when different instances of a service are running on different ports. If the services are equivalent, then it does not matter. However if there are substantial differences in the quality of the service that

information needs to be part of the validation process. If one has just a host name and not a port in the information being validated, then this is not going to be a successful strategy.

3.4. Additional GSS-API Services

GSS-API provides per-message security services that can provide confidentiality and/or integrity. Some IETF protocols such as NFS and SSH take advantage of these services. As a result GSS-EAP needs to support these services. As with mutual authentication, per-message services will limit the set of EAP methods that can be used to those that generate a Master Session Key (MSK). Any EAP method that produces an MSK is able to support per-message security services described in [RFC2743].

GSS-API provides a pseudo-random function. This function generates a pseudo-random sequence using the shared private key as the seed for the bytes generated. This provides an algorithm that both the initiator and acceptor can run in order to arrive at the same key value. The use of this feature allows for an application to generate keys or other shared secrets for use in other places in the protocol. In this regards, it is similar in concept to the TLS extractor (RFC 5705 [RFC5705].). While no current IETF protocols require this, non-IETF protocols are expected to take advantage of this in the near future. Additionally, a number of protocols have found the TLS extractor to be useful in this regards so it is highly probably that IETF protocols may also start using this feature.

4. Privacy Considerations

ABFAB, as an architecture designed to enable federated authentication and allow for the secure transmission of identity information between entities, obviously requires careful consideration around privacy and the potential for privacy violations.

This section examines the privacy related information presented in this document, summarising the entities that are involved in ABFAB communications and what exposure they have to identity information. In discussing these privacy considerations in this section, we use terminology and ideas from [I-D.iab-privacy-considerations].

Note that the ABFAB architecture uses at its core several existing technologies and protocols; detailed privacy discussion around these is not examined. This section instead focuses on privacy considerations specifically related to overall architecture and usage of ABFAB.

4.1. Entities and their roles

In an ABFAB environment, there are four distinct types of entities involved in communication paths. Figure 2 shows the ABFAB architecture with these entity types. We have:

- The client application: usually a piece of software running on a user's device. This communicates with a service (the Relying Party) that the user wishes to interact with.

- The Identity Provider: The home AAA server for the user.

- The Relying Party: The service the user wishes to connect to.

- The federation substrate: A set of entities through which messages pass on their path between RP and AAA server.

As described in detail earlier in this document, when a user wishes to access a Relying Party, a secure tunnel is set up between their client application and their Identity Provider (via the Relying Party and the federation substrate) through which credentials are exchanged. An indication of success or failure, alongside a set of AAA attributes about a principal is then passed from the Identity Provider to the Relying Party (usually in the form of a SAML assertion).

4.2. Relationship between user and entities

- Between User and Identity Provider - the identity Provider is an entity the user will have a direct relationship with, created when the organisation that operates the entity provisioned and exchanged the user's credentials. Privacy and data protection guarantees may form a part of this relationship.

- Between User and Relying Party - the Relying Party is an entity the user may or may not have a direct relationship with, depending on the service in question. Some services may only be offered to those users where such a direct relationship exists (for particularly sensitive services, for example), while some may not require this and would instead be satisfied with basic federation trust guarantees between themselves and the Identity Provider). This may well include the option that the user stays anonymous with respect to the Relying Party (though obviously not to the Identity Provider). If attempting to preserve privacy through the mitigation of data minimisation, then the only attribute information about individuals exposed to the Relying Party should be that which is strictly necessary for the operation of the service.

- Between User and Federation substrate - the user is highly likely to have no knowledge of, or relationship with, any entities involved with the federation substrate (not that the Identity Provider and/or Relying Party may, however). Knowledge of attribute information about individuals for these entities is not necessary, and thus such information should be protected in such a way as to prevent access to this information from being possible.

4.3. Data and Identifiers in use

In the ABFAB architecture, there are a few different types of data and identifiers in use.

4.3.1. NAI

In order for the Relying Party to be able to route messages to enable an EAP transaction to occur between client application and the correct identity Provider, it is necessary for the client application to provide enough information to the Relying Party to enable the identification of the correct Identity Provider. This takes the form of an Network Access Identifier (NAI) (as specified in [RFC4282]). Note that an NAI can have inner and outer forms in a AAA architecture.

- The outer part of NAI is exposed to the Relying Party; this can simply contain realm information. Doing so (i.e. not including user identification details such as a username) minimises the data given to the Relying Part to that which is purely necessary to support the necessary routing decision.

- The inner part of NAI is sent through the secure tunnel as established by the EAP protocol; this form of the NAI will contain credentials for the user suitable for authenticating them successfully (e.g. a username and password). Since the entire purpose of the secure tunnel is to protect communications between client application (EAP client) and Identity Provider (EAP server), then it is considered secure from eavesdroppers or malicious intermediaries and no further privacy discussion is necessary.

4.3.2. Identity Information

As a part of the ABFAB process, after a successful authentication has occurred between client application and Identity Provider, an indication of this success is sent to the Relying Party. Alongside this message, information about the user may be returned through AAA attributes, usually in form of a SAML assertion. This information is arbitrary and may include either only attributes that prevent an individual from being identified by the Relying Party (thus enabling anonymous or pseudonymous access) or attributes that contain personally identifiable information.

Depending on the method used, this information carried through AAA attributes may or may not be accessible to intermediaries involved in communications - e.g. in the case of RADIUS and unencrypted SAML, these headers are plain text and could be seen by any observer, whereas if using RADSEC or encrypted SAML, these headers are protected from observers. Obviously, where the protection of the privacy of an individual is required then this information needs to be protected by some appropriate means.

4.3.3. Accounting Information

Alongside the core authentication and authorization that occurs in AAA communications, accounting information about resource consumption may be delivered as part of the accounting exchange during the lifetime of the granted application session.

4.3.4. Collection and retention of data and identifiers

In cases where Relying Parties do not require to identify a particular individual when an individual wishes to make use of their

service, the ABFAB architecture enable anonymous or pseudonymous access. Thus data and identifiers other than pseudonyms and unlinkable attribute information need not be stored and retained.

However, in cases where Relying Parties require the ability to identify a particular individual (e.g. so they can link this identity information to a particular account in their service, or where identity information is required for audit purposes), the service will need to collect and store such information, and to retain it for as long as they require. Deprovisioning of such accounts and information is out of scope for ABFAB, but obviously for privacy protection any identifiers collected should be deleted when they are no longer needed.

4.4. User Participation

In the ABFAB architecture, by its very nature users are active participants in the sharing of their identifiers as they initiate the communications exchange every time they wish to access a server. They are, however, not involved in control of the set of information related to them that transmitted from Identity Provider to Relying Party for authorisation purposes.

5. Deployment Considerations

5.1. EAP Channel Binding

Discuss the implications of needing EAP channel binding.

5.2. AAA Proxy Behavior

Discuss deployment implications of our proxy requirements.

6. Security Considerations

This document describes the architecture for Application Bridging for Federated Access Beyond Web (ABFAB) and security is therefore the main focus. This section highlights the main communication channels and their security properties:

Client-to-RP Channel:

The channel binding material is provided by any certificates and the final message (i.e., a cryptographic token for the channel). Authentication may be provided by the RP to the client but a deployment without authentication at the TLS layer is possible as well. In addition, there is a channel between the GSS requestor and the GSS acceptor, but the keying material is provided by a "third party" to both entities. The client can derive keying material locally, but the RP gets the material from the IdP. In the absence of a transport that provides encryption and/or integrity, the channel between the client and the RP has no ability to have any cryptographic protection until the EAP authentication has been completed and the MSK is transfered from the IdP to the RP.

RP-to-IdP Channel:

The security of this communication channel is mainly provided by the functionality offered via RADIUS and Diameter. At the time of writing there are no end-to-end security mechanisms standardized and thereby the architecture has to rely on hop-by-hop security with trusted AAA entities or, as an alternative but possible deployment variant, direct communication between the AAA client to the AAA server. Note that the authorization result the IdP provides to the RP in the form of a SAML assertion may, however, be protected such that the SAML related components are secured end-to-end.

The MSK is transported from the IdP to the RP over this channel. As no end-to-end security is provided by AAA, all AAA entities on the path between the RP and IdP have the ability to eavesdrop if no additional security measures are taken. One such measure is to use a transport between the client and the IdP that provides confidentiality.

Client-to-IdP Channel:

This communication interaction is accomplished with the help of EAP and EAP methods. The offered security protection will depend on the EAP method that is chosen but a minimum requirement is to

offer mutual authentication, and key derivation. The IdP is responsible during this process to determine that the RP that is communication to the client over the RP-to-IdP channel is the same one talking to the IdP. This is accomplished via the EAP channel binding.

Partial list of issues to be addressed in this section: Privacy, SAML, Trust Anchors, EAP Algorithm Selection, Diameter/RADIUS/AAA Issues, Naming of Entities, Protection of passwords, Channel Binding, End-point-connections (TLS), Proxy problems

When a psuedonym is generated as a unique long term identifier for a Subject by an IdP, care MUST be taken in the algorithm that it cannot easily be reverse engineered by the service provider. If it can be reversed then the service provider can consult an oracle to determine if a given unique long term identifier is associated with a different known identifier.

7. IANA Considerations

This document does not require actions by IANA.

8. Acknowledgments

We would like to thank Mayutan Arumaithurai and Klaas Wierenga for their feedback. Additionally, we would like to thank Eve Maler, Nicolas Williams, Bob Morgan, Scott Cantor, Jim Fenton, Paul Leach, and Luke Howard for their feedback on the federation terminology question.

Furthermore, we would like to thank Klaas Wierenga for his review of the pre-00 draft version.

9. References

9.1. Normative References

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.

[RFC4072] Eronen, P., Hiller, T., and G. Zorn, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, August 2005.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, December 2005.

[I-D.ietf-abfab-gss-eap] Hartman, S. and J. Howlett, "A GSS-API Mechanism for the Extensible Authentication Protocol", draft-ietf-abfab-gss-eap-09 (work in progress), August 2012.

[I-D.ietf-abfab-aaa-saml] Howlett, J. and S. Hartman, "A RADIUS Attribute, Binding and Profiles for SAML", draft-ietf-abfab-aaa-saml-04 (work in progress), October 2012.

[RFC6677] Hartman, S., Clancy, T., and K. Hoeper, "Channel-Binding Support for Extensible Authentication Protocol (EAP) Methods", RFC 6677, July 2012.

9.2. Informative References

[RFC2903] de Laat, C., Gross, G., Gommans, L., Vollbrecht, J., and D. Spence, "Generic AAA Architecture", RFC 2903,

August 2000.

[I-D.nir-tls-eap]
Nir, Y., Sheffer, Y., Tschofenig, H., and P. Gutmann, "A Flexible Authentication Framework for the Transport Layer Security (TLS) Protocol using the Extensible Authentication Protocol (EAP)", draft-nir-tls-eap-13 (work in progress), December 2011.

[I-D.ietf-oauth-v2]
Hardt, D., "The OAuth 2.0 Authorization Framework", draft-ietf-oauth-v2-31 (work in progress), August 2012.

[I-D.iab-privacy-considerations]
Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", draft-iab-privacy-considerations-03 (work in progress), July 2012.

[I-D.perez-radext-radius-fragmentation]
Perez-Mendez, A., Lopez, R., Pereniguez-Garcia, F., Lopez-Millan, G., Lopez, D., and A. DeKok, "Support of fragmentation of RADIUS packets", draft-perez-radext-radius-fragmentation-05 (work in progress), February 2013.

[RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, March 2005.

[RFC5106] Tschofenig, H., Kroeselberg, D., Pashalidis, A., Ohba, Y., and F. Bersani, "The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method", RFC 5106, February 2008.

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, June 1996.

[RFC2203] Eisler, M., Chiu, A., and L. Ling, "RPCSEC_GSS Protocol Specification", RFC 2203, September 1997.

[RFC3645] Kwan, S., Garg, P., Gilroy, J., Esibov, L., Westhead, J., and R. Hall, "Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG)", RFC 3645, October 2003.

[RFC2138] Rigney, C., Rigney, C., Rubens, A., Simpson, W., and S.

Willens, "Remote Authentication Dial In User Service (RADIUS)", RFC 2138, April 1997.

[RFC4462] Hutzelman, J., Salowey, J., Galbraith, J., and V. Welch, "Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol", RFC 4462, May 2006.

[RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, November 2007.

[RFC5080] Nelson, D. and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", RFC 5080, December 2007.

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, March 2010.

[RFC5801] Josefsson, S. and N. Williams, "Using Generic Security Service Application Program Interface (GSS-API) Mechanisms in Simple Authentication and Security Layer (SASL): The GS2 Mechanism Family", RFC 5801, July 2010.

[RFC5849] Hammer-Lahav, E., "The OAuth 1.0 Protocol", RFC 5849, April 2010.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, May 2012.

[OASIS.saml-core-2.0-os]
Cantor, S., Kemp, J., Philpott, R., and E. Maler, "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard saml-core-2.0-os, March 2005.

[RFC2904] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M., and D. Spence, "AAA Authorization Framework", RFC 2904, August 2000.

[I-D.ietf-emu-crypto-bind]
Hartman, S., Wasserman, M., and D. Zhang, "EAP Mutual Cryptographic Binding", draft-ietf-emu-crypto-bind-02 (work in progress), February 2013.

[I-D.ietf-emu-eap-tunnel-method]
Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel EAP Method (TEAP) Version 1", draft-ietf-emu-eap-tunnel-method-05 (work in progress), February 2013.

[I-D.ietf-radext-dtls]
DeKok, A., "DTLS as a Transport Layer for RADIUS", draft-ietf-radext-dtls-03 (work in progress), January 2013.

[WS-TRUST]
Lawrence, K., Kaler, C., Nadalin, A., Goodner, M., Gudgin, M., Barbir, A., and H. Granqvist, "WS-Trust 1.4", OASIS Standard ws-trust-200902, February 2009, <http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html>.

[NIST-SP.800-63]
Burr, W., Dodson, D., and W. Polk, "Electronic Authentication Guideline", NIST Special Publication 800-63, April 2006.

URIs

[1] <http://www.openid.net>

[2] <http://www.eduroam.org>

Editorial Comments

[anchor4] JLS: Should this be an EAP failure to the client as well?

[anchor7] JLS: I don't believe this is a true statement - check it with Josh and Sam.

Authors' Addresses

Josh Howlett
JANET(UK)
Lumen House, Library Avenue, Harwell
Oxford OX11 0SG
UK

Phone: +44 1235 822363
Email: Josh.Howlett@ja.net

Sam Hartman
Painless Security

Phone:
Email: hartmans-ietf@mit.edu

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland

Phone: +358 (50) 4871445
Email: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at

Eliot Lear
Cisco Systems GmbH
Richtistrasse 7
Wallisellen, ZH CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Jim Schaad
Soaring Hawk Consulting

Email: ietf@augustcellars.com