ABFAB
Internet-Draft
Intended status: Informational
Expires: November 25, 2012

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

May 24, 2012

Application Bridging for Federated Access Beyond Web (ABFAB) Architecture
draft-ietf-abfab-arch-02.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 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 November 25, 2012.

Copyright Notice

Table of Contents

1. Introduction 4
1.1. Terminology 5
1.2. An Overview of Federation 6
1.3. Challenges to Contemporary Federation 9
1.4. An Overview of ABFAB-based Federation 9
1.5. Design Goals 12
1.6. Client to Relying Party Transport 13
1.7. Use of AAA 14
2. Architecture 15
2.1. Relying Party to Identity Provider 16
2.2. Client To Identity Provider 19
2.3. Client to Relying Party 20
3. Application Security Services 23
3.1. Authentication 23
3.2. GSS-API Channel Binding 24
3.3. Host-Based Service Names 25
3.4. Per-Message Tokens 26
4. Future Work: Attribute Providers 27
5. Privacy Considerations 28
5.1. What Entities collect and use Data? 28
5.2. Relationship between User's and other Entities 29
5.3. What Data about the User is likely Needed to be Collected? 29
5.4. What is the Identification Level of the Data? 29
5.5. Privacy Challenges 30
6. Deployment Considerations 31
6.1. EAP Channel Binding 31
6.2. AAA Proxy Behavior 31
7. Security Considerations 32
8. IANA Considerations 34
9. Acknowledgments 35
10. References 36
10.1. Normative References 36
10.2. Informative References 36
Authors' Addresses 40

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], 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 Subject. Many security mechanisms are manifested as an exchange of information between these actors. The RP is therefore able to decide whether the Subject is authorised, 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 such standards as 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 may also therefore require fewer credentials, which is often desirable.

Privacy:

> 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 to establish, for example, that the Subject is affiliated with a particular organisation or has a certain role or entitlement. Sometimes the RP does require an identifier for the Subject (for example, so that it can

recognise the Subject subsequently); in this case, it is a common practise for the IdP to only release a pseudonym that is specific to that particular Relying Party. Federated access management therefore provides various strategies for protecting the Subject's privacy. Other privacy aspects typically of concern are the policy for releasing personal data about the Subject from the IdP to the RP, the purpose of the usage, the retention period of the data, and many more.

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 protocol, the Generic Security Service (GSS), the GS2 family, the Extensible Authentication Protocol (EAP) and 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.

1.1. Terminology

This document uses identity management and privacy terminology from [I-D.iab-privacy-terminology]. In particular, this document uses the terms identity provider, relying party, (data) subject, 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].

One of the problems people will find with reading this document is that the terminology somestimes appears to be inconsistant. This is do 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 consistant 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	Subject	Relying Party (RP)	Identity Provider (IdP)
SAML	Principal		
GSS-API			
EAP	EAP client		EAP server
SASL	EAP peer		
AAA		AAA Client	AAA server
RADIUS	client	NAS	RADIUS server

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.

1.2. An Overview of Federation

In the previous section we introduced the following actors:

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

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

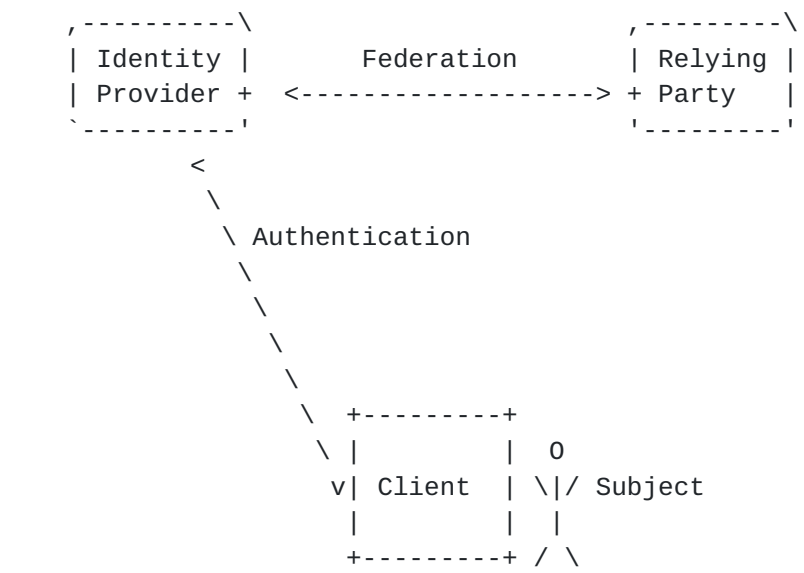


Figure 1: Entities and their Relationships

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 these specifications to make the system work and to accomplish interoperability.

Legal Rules:

The legal rules take existing laws into consideration and provide contractual obligations to provide further clarification and define responsibilities. 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. These legal rules may, for example, describe liability for losses, termination rights, enforcement mechanisms, measures of damage, dispute resolution, warranties, etc.

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.

While many of the non-technical aspects of federation, such as business practices and legal arrangements, are outside the scope of the IETF they still impact the architectural setup on how to ensure the dynamic establishment of trust.

Some deployments demand the deployment of sophisticated technical infrastructure, including message routing intermediaries, to offer the required technical functionality. In other deployments fewer technical components are needed.

Figure 1 also shows the relationship between the IdP and the Subject. Often a real world entity is associated with the Subject; for example, a person or some software.

The IdP will typically have a long-term relationship with the Subject. This relationship would typically involve the IdP positively identifying and credentialling the Subject (for example, at time of enrollment in the context of employment within an organisation). The relationship will often be instantiated within an agreement between the IdP and the Subject (for example, within an employment contract or terms of use that stipulates the appropriate use of credentials and so forth).

While federation is often discussed within the context of relatively formal relationships, such as between an enterprise and an employee or a government and a citizen, federation does not in any way require this; nor, indeed, does it require any particular level of formality. It is, for example, entirely compatible with a relationship between the IdP and Subject that is only as weak as completing a web form and confirming the verification email.

However, 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 Subject made by an IdP. This is sometimes described as the Level of Assurance.

Similarly it is also important to note that, in the general case, there is no requirement of a long-term relationship between the RP and the Subject. This is a property of federation that yields many

of its benefits. However, federation does not preclude the possibility of a pre-existing relationship existing between the RP and the Subject, 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 a human behind the device denoted as Client and in other cases there is no human involved in the actual protocol execution.

1.3. Challenges to 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 modifying the grades. A teacher who is also a parent must clearly distinguish here 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 does not on its own provide for a way to select a federation. 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.

The steps taken in an ABFAB federated authentication/authorization exchange are as follows:

1. Principal provides an NAI to Application: The client application is configured with an NAI. The provided name contains, at a minimum, the realm of an NAI. The realm represents the IdP to be discovered.

2. Authentication mechanism selection: The GSS-EAP SASL/GS2 mechanism is selected for authentication/authorization.

3. Client Application provides the NAI to RP: The client application setups a transport to the RP and begins the GSS-EAP

authentication. The RP initiates the EAP protocol to the client application, and the client provides the NAI to the RP in the form of the EAP response.

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 provided NAI. This is discussed in detail below.

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 principal is. The RP claims its identity to the IdP in AAA attributes, and it may contain a SAML Attribute Requests in a AAA attribute.

6. IdP informs the principal of which EAP method to use: The available and appropriate methods are discussed below in this memo.

7. The EAP protocol is run: A bunch of EAP messages are passed between the EAP peer and the EAP server, i.e., the principal and the IdP in our identity management terminology, until the result of the authentication protocol is determined. The number and content of those messages will depend on the EAP method. If the IdP is unable to authenticate the principal, the process concludes here. As part of the EAP protocol, the principal will create a channel bindings message to the IdP identifying, among other things, the RP to which it is attempting to authenticate. The IdP checks the channel binding data from the principal with that provided by the RP via the AAA protocol. If the bindings do not match the IdP fails the EAP protocol.

8. Successful Authentication: The IdP (its EAP server) and EAP peer / subject have mutually authenticated each other. As a result of this step, the subject and the IdP hold two cryptographic keys- a Master Session Key (MSK), and an Extended MSK (EMSK). There is some confidence that the RP is the one the principal is communicating with as the channel binding data validated. This allows for a claim of authentication for the RP to the principal.

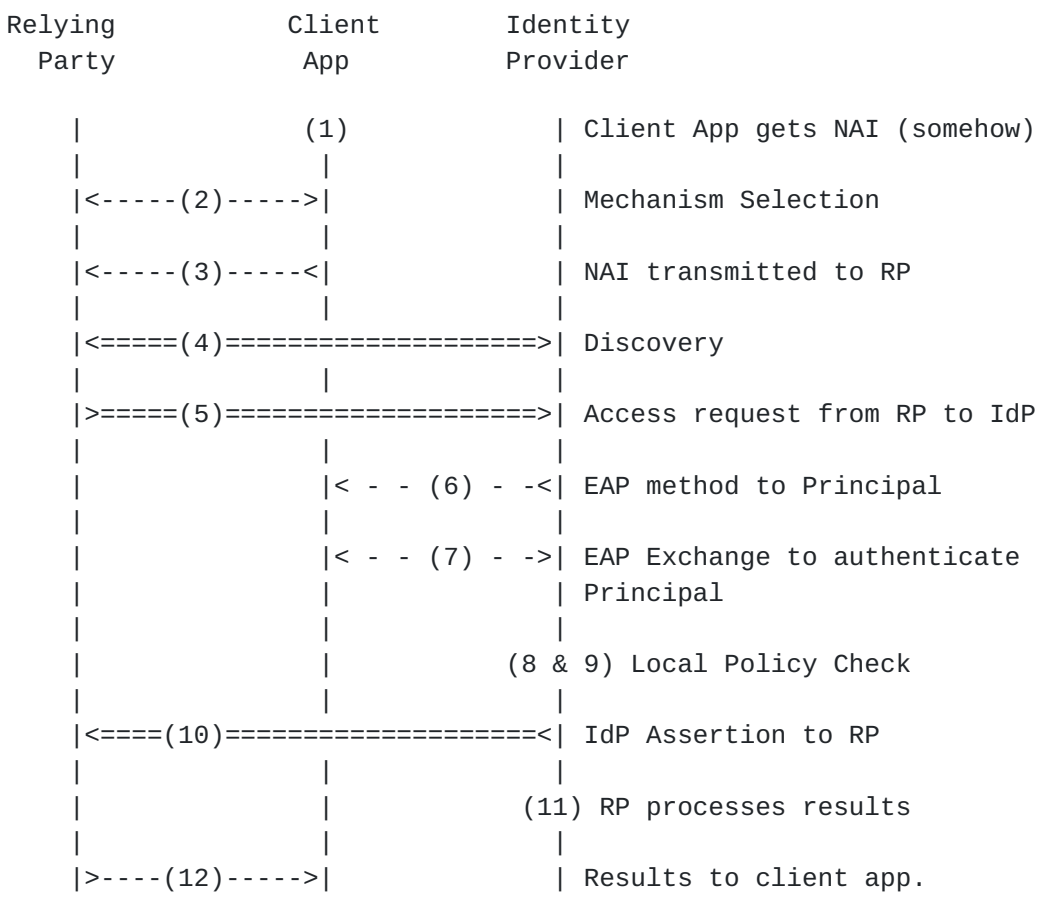
9. Local IdP Policy Check: At this stage, the IdP checks local policy to determine whether the RP and subject are authorized for a given transaction/service, and if so, what if any, attributes will be released to the RP. The RP will have done it's policy checks during the discovery process.

10. Response from the IdP to the Relying Party: Once the IdP has made a determination of whether and how to authenticate and authorize the principal to the RP, it returns either a negative AAA result to the RP, or it returns a positive result to the RP, along with an optional set of AAA attributes associated with the principal that could include one or more SAML assertions. In addition, an 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 principal 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 principal: 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.

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, and the principal 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.

- 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 frought 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.

1.6. Client to Relying Party Transport

The transport of data between the client and the relying part 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.

1.7. Use of AAA

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].

Is it possible to design a system that builds on top of successful protocols to offer non-Web-based protocols with a solid starting point for authentication and authorization in a distributed system?

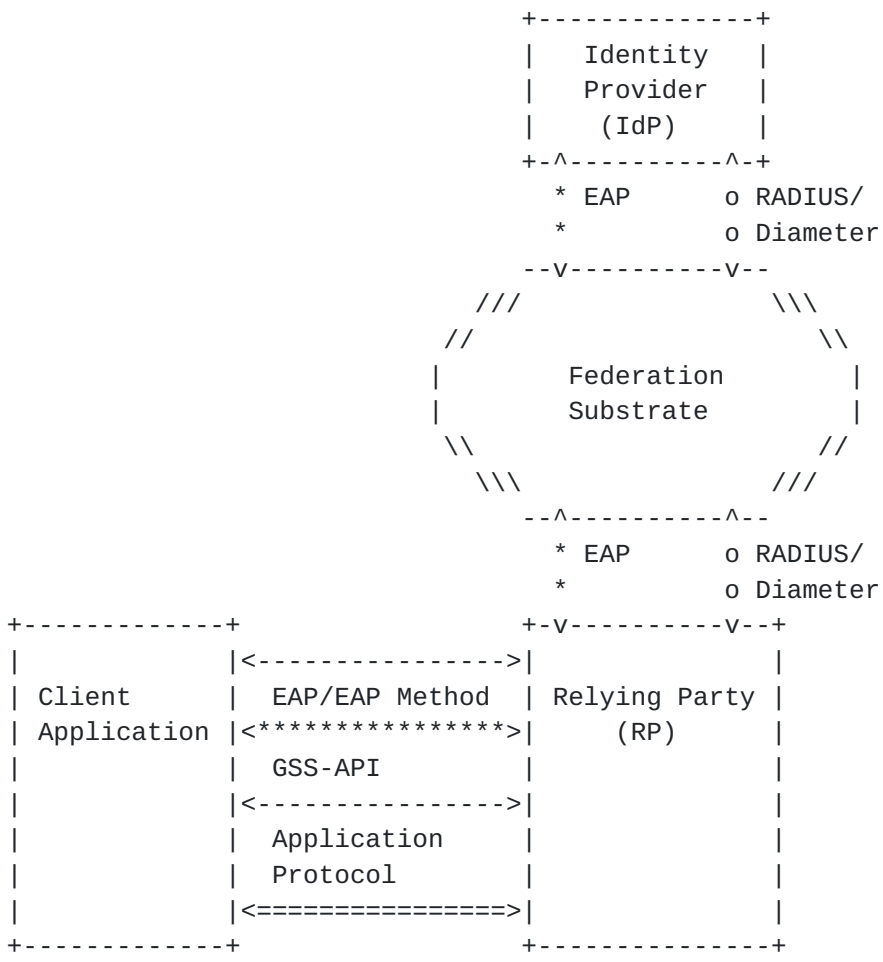
2. Architecture

Section 1 already introduced the federated access architecture, with the illustration of the different actors that need to interact, but it 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 both the relying party as well as to the client for application integration, 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. Encapsulating EAP into the GSS-API also allows EAP to be used in SASL. 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].

The architecture consists of several buiding blocks, which is shown graphically in Figure 2. The subsections below explain relationship of the protocol components in more detail.



Legend:

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

Figure 2: ABFAB Protocol Instantiation

2.1. Relying Party to Identity Provider

The federation substrate is responsible for the communication between the relying party and the identity provider. This layer is responsible for the inter-domain communication and for the technical mechanisms necessary to establish inter-domain trust.

A key design goal is the re-use of an existing infrastructure, we build upon the AAA framework as utilized by RADIUS [RFC2138] and Diameter [RFC3588]. Since this document does not aim to re-describe the AAA framework the interested reader is referred to [RFC2904].

Building on the AAA infrastructure, and RADIUS and Diameter as protocols, modifications to that infrastructure is to be avoided. Also, modifications to AAA servers should be kept at a minimum.

The astute reader will notice that RADIUS and Diameter have substantially similar characteristics. Why not pick one? A key difference is that today RADIUS is largely transported upon UDP, and its use is largely, though not exclusively, intra-domain. Diameter itself was designed to scale to broader uses. We leave as a deployment decision, which protocol will be appropriate.

Through the integrity protection mechanisms in the AAA framework, the relying party 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 is 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 principal on its own, without the assistance of an IdP. This architecture makes use of the Network Access Identifier (NAI), where the IdP is indicated in 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.

The RP needs to discover which federation will be used to contact the IDP. As part of this process, the RP determines 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 intermediates validate information that is being communicated within the federation.

Several deployment approaches are possible. These 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 constrain the names of both parties to the exchange. The trust mechanism ought 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. Naming 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 needed for an RP to discover what endpoint to contact for a given NAI. Certificates provide a place to store information about rules determination and naming constraints. Provided that no intermediates are 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 MAY use any of the three methods to get closer to the IDP. It is also likely that rather than being directly reachable, an IDP may have a proxy within its organization. Federations MAY provide a traditional AAA proxy interface even if they also provide another mechanism for increased efficiency or security.

2.2. Client To Identity Provider

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. It is difficult to have subjects that are machines rather than humans that use some sort of programatic credential. In addition, 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: one is 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.

2.3. Client to Relying Party

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.

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 supplied by 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 supplied by the initiator, then the initiator trusts that its idea of the acceptor name 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, users 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 acceptor 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 capabale of providing the necessary service can be used, and appropriate steps need to be taken to provide mutual authentication. A broader set of EAP methods could be supported when a particular application does not request mutual authentication. It is an open question whether the mechanism will permit this.

The AAA infrastructure MAY hide the peer's identity from the GSS-API acceptor, providing anonymity between the peer and initiator. At this time, whether the identity is disclosed is determined by EAP server policy rather than by an indication from the peer. Also, peers are unlikely to be able to determine whether anonymous communication will be provided. For this reason, peers 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 prevent man-in-the-middle attacks. It is common to provide SASL and GSS-API with another layer to provide transport security; Transport Layer Security (TLS) is the most common such layer. 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. 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. XXXThere is an open question here as to the details; today RFC 5554 governs. We could use that and the current draft assumes we will. However in Beijing we became aware of some changes to these details that would make life much better for GSS authentication of HTTP. We should resolve this with kitten and replace this note with a reference to the spec we're actually

following.

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.

RFC 5056 channel binding (also called GSS-API channel binding when GSS-API is involved) is not the same thing as EAP channel binding. EAP channel binding is also used in the ABFAB context in order to implement acceptor naming and mutual authentication. Details are discussed in the mechanisms specification [I-D.ietf-abfab-gss-eap].

3.3. Host-Based Service Names

IETF security mechanisms typically take the name of a service entered by a user and make some trust decision about whether the remote party in an interaction is the intended party. GSS-API has a relatively flexible naming architecture. However most of the IETF applications that use GSS-API, including SSH, NFS, IMAP, LDAP and XMPP, have chosen to use host-based service names when they use GSS-API. In this model, the initiator names an acceptor based on a service such as "imap" or "host" (for login services such as SSH) and a host name.

Using host-based service names leads to a challenging trust delegation problem. Who is allowed to decide whether a particular hostname maps to an entity. The public-key infrastructure (PKI) used by the web has chosen to have a number of trust anchors (root certificate authorities) each of wich can map any name to a public key. A number of GSS-API mechanisms suchs as Kerberos [RFC1964] split the problem into two parts. A new concept called a realm is introduced. Then the mechanism decides what realm is responsible for a given name. That realm is responsible for deciding if the acceptor

entity is allowed to claim the name. ABFAB needs to adopt this approach.

Host-based service names do not work ideally when different instances of a service are running on different ports. Also, these do not work ideally when SRV record or other insecure referrals are used.

The GSS-EAP mechanism needs to support host-based service names in order to work with existing IETF protocols.

3.4. Per-Message Tokens

GSS-API provides per-message security services that can provide confidentiality and 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 are available. Any method that produces a Master Session Key (MSK) should be able to support per-message security services.

GSS-API provides a pseudo-random function. While the pseudo-random function does not involve sending data over the wire, it provides an algorithm that both the initiator and acceptor can run in order to arrive at the same key value. This is useful for designs where a successful authentication is used to key some other function. This is similar in concept to the TLS extractor. No current IETF protocols require this. However GSS-EAP supports this service because it is valuable for the future and easy to do given per-message services. Non-IETF protocols are expected to take advantage of this in the near future.

4. Future Work: Attribute Providers

This architecture provides for a federated authentication and authorization framework between IdPs, RPs, principals, and subjects. It does not at this time provide for a means to retrieve attributes from 3rd parties. However, it envisions such a possibility. We note that in any extension to the model, an attribute provider must be authorized to release specific attributes to a specific RP for a specific principal. In addition, we note that it is an open question beyond this architecture as to how the RP should know to trust a particular attribute provider.

There are a number of possible technical means to provide attribute provider capabilities. One possible approach is for the IdP to provide a signed attribute request to RP that it in turn will provide to the attribute authority. Another approach would be for the IdP to provide a URI to the RP that contains a token of some form. The form of communications between the IdP and attribute provider as well as other considerations are left for the future. One thing we can say now is that the IdP would merely be asserting who the attribute authority is, and not the contents of what the attribute authority would return. (Otherwise, the IdP might as well make the query to the attribute authority and then resign it.)

5. Privacy Considerations

Sharing identity information raises privacy violations and as described throughout this document an existing architecture is re-used for a different usage environment. As such, a discussion about the privacy properties has to take these pre-conditions into consideration. We use the approach suggested in [I-D.iab-privacy-considerations] to shed light into what data is collected and used by which entity, what the relationship between these entities and the end user is, what data about the user is likely needed to be collected, and what the identification level of the data is.

5.1. What Entities collect and use Data?

Figure 2 shows the architecture with the involved entities. Message exchanges are exchanged between the client application, via the relying part to the AAA server. There will likely be intermediaries between the relying party and the AAA server, labeled generically as "federation".

In order for the relying party to route messages to the AAA server it is necessary for the client application to provide enough information to enable the identification of the AAA server's domain. While often the username is attached to the domain (in the form of a Network Access Identity (NAI) this is not necessary for the actual protocol operation. The EAP server component within the AAA server needs to authenticate the user and therefore needs to execute the respective authentication protocol. Once the authentication exchange is complete authorization information needs to be conveyed to the relying party to grant the user the necessary application rights. Information about resource consumption may be delivered as part of the accounting exchange during the lifetime of the granted application session.

The authentication exchange may reveal an identifier that can be linked to a user. Additionally, a sequence of authentication protocol exchanges may be linked together. Depending on the chosen authentication protocol information at varying degrees may be revealed to all parties along the communication path. This authorization information exchange may disclose identity information about the user. While accounting information is created by the relying party it is likely to needed by intermediaries in the federation for financial settlement purposes and will be stored for billing, fraud detection, statistical purposes, and for service improvement by the AAA server operator.

5.2. Relationship between User's and other Entities

The AAA server is a first-party site the user typically has a relationship with. This relationship will be created at the time when the security credentials are exchange and provisioned. The relying party and potential intermediares in the federation are typically operate under the contract of the first-party site. The user typically does not know about the intermediaries in the federation nor does he have any relationship with them. The protocol interaction triggered by the client application happens with the relying party at the time of application access. The relying party does not have a direct contractual relationship with the user but depending on the application the interaction may expose the brand of the application running by the relying party to the end user via some user interface.

5.3. What Data about the User is likely Needed to be Collected?

The data that is likely going to be collected as part of a protocol exchange with application access at the relying party is accounting information and authorization information. This information is likely to be kept beyond the terminated application usage for trouble shooting, statistical purposes, etc. There is also like to be additional data collected to to improve application service usage by the relying party that is not conveyed to the AAA server as part of the accounting stream.

5.4. What is the Identification Level of the Data?

With regard to identification there are several protocol layers that need to be considered separately. First, there is the EAP method exchange, which as an authentication and key exchange protocol has properties related to identification and protocol linkage. Second, there is identification at the EAP layer for routing of messages. Then, in the exchange between the client application and the relying party the identification depends on the underlying application layer protocol the EAP/GSS-API exchange is tunneled over. Finally, there is the backend exchange via the AAA infrastructure, which involves a range of RADIUS and Diameter extensions and yet to be defined extensions, such as encoding authorization information inside SAML assertions.

Since this document does not attempt to define any of these exchanges but rather re-uses existing mechanisms the level of identification heavily depends on the selected mechanisms. The following two examples aim to illustrate the amount of existing work with respect to decrease exposure of personal data.

1. When designing EAP methods a number of different requirements may need to get considered; some of them are conflicting. RFC 4017 [RFC4017], for example, tried to list requirements for EAP methods utilized for network access over Wireless LANs. It also recommends the end-user identity hiding requirement, which is privacy-relevant. Some EAP methods, such as EAP-IKEv2 [RFC5106], also fulfill this requirement.

2. EAP, as the layer encapsulating EAP method specific information, needs identity information to allow routing requests towards the user's home AAA server. This information is carried within the Network Access Identifier (NAI) and the username part of the NAI (as supported by RFC 4282 [RFC4282]) can be obfuscated.

5.5. Privacy Challenges

While a lot of standarization work was done to avoid leakage of identity information to intermediaries (such as eavesdroppers on the communication path between the client application and the relying party) in the area of authentication and key exchange protocols. However, from current deployments the weak aspects with respect to security are:

1. Today business contracts are used to create federations between identity providers and relying parties. These contracts are not only financial agreements but they also define the rules about what information is exchanged between the AAA server and the relying party and the potential involvement of AAA proxies and brokers as intermediaries. While these contracts are openly available for university federations they are not public in case of commercial deployments. Quite naturally, there is a lack of transparency for external parties.

2. In today's deployments the ability for users to determine the amount of information exchanged with other parties over time, as well as the possibility to control the amount of information exposed via an explict consent is limited. This is partially due the nature of application capabilities at the time of network access authentication. However, with the envisioned extension of the usage, as described in this document, it is desirable to offer these capabilities.

6. Deployment Considerations

6.1. EAP Channel Binding

Discuss the implications of needing EAP channel binding.

6.2. AAA Proxy Behavior

Discuss deployment implications of our proxy requirements.

7. 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 fis 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

8. IANA Considerations

This document does not require actions by IANA.

9. 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, 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.

10. References

10.1. 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.iab-privacy-terminology] Hansen, M., Tschofenig, H., Smith, R., and A. Cooper, "Privacy Terminology and Concepts", draft-iab-privacy-terminology-01 (work in progress), March 2012.

[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-07 (work in progress), May 2012.

10.2. Informative References

[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]
Hammer-Lahav, E., Recordon, D., and D. Hardt, "The OAuth 2.0 Authorization Framework", draft-ietf-oauth-v2-26 (work in progress), May 2012.

[I-D.iab-privacy-considerations]
Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., and J. Morris, "Privacy Considerations for Internet Protocols", draft-iab-privacy-considerations-02 (work in progress), March 2012.

[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.

[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.

[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.

[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>.

URIs

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

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

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