ABFAB R. Smith
Internet-Draft Cardiff University
Intended status: Informational M. Donnelly
Expires: September 22, 2016 Painless Security
March 21, 2016

Application Bridging for Federated Access Beyond web (ABFAB) Usability and User Interface Considerations
draft-ietf-abfab-usability-ui-considerations-04

Abstract

The real world use of ABFAB-based technologies requires that any identity that is to be used for authentication has to be configured on the ABFAB-enabled client device. Achieving this requires software on that device (either built into the operating system or a standalone utility) that will interact with the user, managing their identity information and identity-to-service mappings. All designers of software to fulfil this role will face the same set of challenges. This document aims to document these challenges with the aim of producing well-thought out UIs with some degree of consistency between implementations.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The use of ABFAB-based technologies requires that any identity that is to be used for authentication has to be configured on the client device. Achieving this requires software on that device (either built into the operating system or a standalone utility) that will interact with the user, and manage the user’s identities and credential-to-service mappings. Anyone designing that software will face the same set of challenges.

This document does not intend to supplant evidence-based UI design guidelines; implementers of identity selectors are strongly encouraged to understand the latest in HCI and UX thought and practice. Instead, it aims to document the common challenges faced by implementers with the aim of providing a common starting point for implementers in the hope that this aids in producing well-thought out UIs with some degree of consistency.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Terminology

Various items of terminology used in the document are heavily overloaded in that they mean a variety of different things to different people. In an attempt to minimise this problem, this section gives a brief description of the main items of terminology used in order to aid a consistent understanding of this document.

- NAI: Network Access Identifier - a standard way of identifying a user and assisting in the routing of an authentication request (see [RFC4282]).

- Identity: In this context, an identity is a credential given to a user by a particular organisation with which they have an association. A user may have multiple identities - potentially multiple identities per organisation, and also across multiple organisations. Each identity will consist of an NAI, alongside other information that supports authentication. Note that in other contexts the usual use of "identity" would match our use of "user", whereas the usual use of "identifier" matches our use of identity.

- Service: The thing that the user is attempting to authenticate to via ABFAB technology. See [I-D.ietf-abfab-usecases] for some example ABFAB use cases. Also known as the Relying Party.

- Identity Provider: The thing able to make access management decisions about the Identity.

- Identity Selector: A piece of software that enables the process by which the GSS-API acquires the identity to use with a particular service. An Identity Selector typically would allow the user to configure a set of identities along with service to identity mappings.

- Trust anchor: An authoritative source of verification of a particular ABFAB Identity Provider, used to allow authentication of an Identity Provider using X.509 [RFC5280]. Typically this will be a commercial CA to allow authentication via chain of trust, or a preconfigured non-commercial certificate (e.g. self-signed).

- Credential: Whatever is used by the user to authenticate themselves with a particular NAI. What exactly this will be will be dependent on the EAP method being used, but is likely to be something like a password or a certificate.

4. Context

When using the ABFAB architecture (see [I-D.ietf-abfab-arch]) to perform federated authentication to some service, a user will need to provide identity information that they wish to use to authenticate to that particular service. This will happen through a process of the application calling the GSS-API, which will in turn gather the user’s credentials through some process. We will call this process the "identity selector" in this document (though note that this is not a recommendation on terminology for the process).

The simplest way to achieve the desired effect would be a process that simply takes the credentials from the currently logged in user (e.g. the Windows Domain Credentials) and uses those for all services that request authenticate through ABFAB. This approach gives ultimate simplicity in terms of UI (it wouldn’t have one) but the least flexibility (the user has to use a single identity for everything). If there is ever to be a requirement for a user to use a different set of credentials for a service, or a requirement for the user to use ABFAB to authenticate to the operating system, then something more complex will be needed.

Where there is a requirement for multiple credentials to be supported, there are at least two methods that could be employed to configure identities and associated information:

- They could be configured manually by the user in a configuration file that could be edited by hand or some such simple process, and read by the GSS-API mechanism. While this could work very well functionally, in practice only a small subset of users would be happy with - and able to - configure their identities in such a manner.

- They could be configured through some interactive process. For ease of use this should have a simple UI, although to support some use cases a headless mode (i.e. a way of interacting with the identity selector when there is no GUI present) may need to be supported.

When designing an identity selector with a UI (or indeed, with a headless mode), any implementer will share a common set of usability considerations inherent to the context. This document aims to explore these considerations, and provide advice and guidance on addressing them where possible.

5. Considerations around Terminology

Anyone designing an identity selector will have to grapple with choosing terminology that the average user has some chance of understanding. This terminology can split into a few main functional areas, as discussed next.

5.1. Identity

The first area where terminology is needed is around the identity/identities of the user. Users are typically used to seeing a variety of terms for aspects of their identity in the federated sense, and an even larger variety in the wider Internet sense. For example, in the federated sense some of these terms include "username", "login", "network account", "institutional account", "home organisation account", "credentials", and a myriad of other such terms. However, NAI - the technically correct name for their identity in an ABFAB sense - is highly unlikely to be one of these terms that users are used to seeing. Further, given that the NAI superficially looks like an email address, there is a definite potential for confusion.

Implementers of an identity selector will need to carefully consider their intended audience for both their level of technical capability and the existing terminology that they may have been exposed to.

Beyond terminology, careful thought needs to be given to the paradigm to use when presenting identity to users, as identities and services are abstract concepts that some users may not find easily understandable. Implementers may wish to keep such abstract concepts despite this, or may wish to examine attempts to map to real world paradigms, e.g. the idea of using "Identity Cards" that are held in the user’s "Wallet", as used by the now defunct Microsoft Cardspace ([MS-CS]).

5.2. Services

Terminology around services is likely to be less of a problem than identity, but it will actually depend on what the service is. For example, each service could be simply described as "server", "system", etc. But for simplicity just the word "service" will probably usually suffice.

5.3. Identity to Service Mapping

The basic functionality of the Identity Selector is to create the correct combination of Identity and Service, so that the correct identity is chosen to create the credential for the GSS-EAP connection with the given service. Mapping is the process of creating this relationship between identity and service.

Depending on your perspective either each identity may be mapped to multiple services, or each service has multiple identities mapped to it. Thus any UI could present either perspective, or both.

6. Considerations around Management of Identities

One of the core features of an identity selector is the management of a user’s identities. This section first looks at what information associated with an identity will need to manage, and then looks in detail at various usability considerations of this area.

6.1. Information associated with each Identity

The bare minimum set of information that MUST be stored about each identity to allow ABFAB authentication to take place is a single item:

- NAI: The user’s Network Access Identifier (see [RFC4282]) for this particular credential. For example, "joe@example.com". Note that the identity selector MUST NOT store different identities that use the same NAI. This is required as the NAI is the unique key that is used by the identity selector when interacting with the GSS-API mechanism for various reasons, for example, to allow the GSS-API mechanism to report back error or success statuses or to allow the application to request the use of a specific identity.

Next up is a small set of information that SHOULD be stored about each identity to allow the user to effectively select a particular identity:

- Identity provider realm: The ABFAB realm of the identity provider. This is used as a key to look up the identity provider from the identity selector’s list of identity providers, in order to access the trust anchor during verification of the identity provider.

- Credential: Whatever is used by the users to authenticate themselves with a particular NAI. What exactly this will be will be dependent on the EAP method being used, but is likely to be something like a password or a certificate. Note that the identity selector SHOULD allow a user to store the credential. However, there are use cases where a user may specifically opt for this not to be "remembered", so the identity selector MUST NOT store the credential without confirmation from the user.

Finally, there is a set of optional information that MAY be stored about each identity that represent useful information for the user to have and could make an identity selector more usable. Note that this list is neither intended to be exhaustive or even particularly correct; any implementer is free to use whatever make sense in their implementation and conforms to good HCI/UX guidelines. Instead, it is simply a suggested starting point.

- Friendly name for identity: To allow the user to differentiate between the set of identities represented in the Identity Selector. This should be editable by the user. The only restriction on this name is that it MUST be unique within that particular user’s set of identities. For example: "Student username", "Google Account", "Work Login", etc.

- Friendly icon for identity: To allow the user to differentiate between the set of identities they have they should be able to set an icon for that particular identity.

- Password changing URL: The URL the user should visit should they need to change their password for this particular identity. For example, "http://www.example.com/passwordreset?identifier=myId".

- Helpdesk URL: The URL this particular identity should visit to get contact details for the helpdesk of the organisation that issued this particular identity for when the user encounters issues and needs help. For example, https://www.example.com/helpdesk?identifier=myId.

6.2. Information associated with each Identity Provider

Identity providers are entities that may be shared across multiple identities. For instance, a person at a university may have one identity as a student and another identity as an employee, but a single identity provider makes access management decisions about both. In these cases, the identity selector MUST consider it an error if the trust anchor for the identity provider is different between the various identities managed by the single identity provider.

The bare minimum set of information that MUST be stored about each identity provider is:

- Realm: The realm of the identity provider. This will uniquely identify the identity realm.

- Trust anchor: For the identity selector to be able to verify that the Identity Provider it is going to talk to and attempt to authenticate against is the Identity Provider that it is expecting, and that it is not being spoofed in some way. This is likely to be an X.509 certificate [RFC5280], or a tuple of (trusted root certificate, servername in Subject or subjectAltName). Storing a credential without a relevant trust anchor allows for the possibility of a malicious attacker intercepting traffic and masquerading as the Identity Provider in question.

Identity providers also have a set of optional information that MAY be stored about each identify provider. This set includes, but is not limited to:

- Friendly name for the identity provider: To allow the user to differentiate between the set of identity providers represented in the Identity Selector. This should be editable by the user. The only restriction on this name is that it MUST be unique within that particular user's set of identity providers. For example: "My University", "Google", etc.

- Friendly icon for the identity provider: To allow the user to differentiate between the set of identity providers they have they should be able to set an icon for that particular identity provider.

- Password changing URL: The URL the user should visit should they need to change passwords for identities in this realm. For example, "http://www.example.com/passwordreset".

- Helpdesk URL: The URL the user should visit to get contact details for the helpdesk of the organisation that issued this particular identity for when the user encounters issues and needs help. For example, https://www.example.com/helpdesk.

Note that the password changing URL and helpdesk URL somewhat mirror the definitions of the same fields in the identity. The distinction is that the URLs in the identity SHOULD apply to the individual identity, whereas the URLs in the identity provider SHOULD apply to all identities that the identity provider defines. For example, an identity password change URL would provide a personalized experience of changing the password for the given identity, but the identity provider password change URL would direct the user to a page where the user would need to enter the individual identity that needs a new password.

If the identity contains no password change URL or helpdesk URL, the identity selector MAY present any corresponding URL from the identity selector instead. However, if the identity contains the URL, the identity selector SHOULD present the URL from the identity.

6.3. Storage of Identity Information

Since some of the information that makes up the identity is sensitive in nature (e.g. containing passwords), then this information SHOULD be stored and accessed securely. This might involve ensuring the credential information is held in encrypted form on device and accessed using a passphrase. For deeper integration into the system, this could be done by using existing secure storage on the system such as Keychain on a Mac, the GNOME keyring on a GNOME based Linux device, or the Credentials Manager on Windows.

6.4. Adding/Association of an Identity

Users will have one or more identities given to them by organisations that they have a relationship with. One of the core tasks of an identity selector will be to learn about these identities and their identity providers in order to use them when it comes to authenticating to services on behalf of the user. Adding these identities could be done in one of three ways: manual addition, automated addition that is manually triggered, or automated addition that is automatically triggered. Each of these are discussed in more detail next.

Note that the term "association" or "addition" of an identity is used rather than "provisioning" of an identity, because while we actually are provisioning identities into the UI, provisioning is an overloaded term in the identity and access management space and could easily be confused with identity provisioning in the sense of the creation of the identity by the home organisation’s identity management procedures.

6.4.1. Identity Provider Addition

6.4.1.1. Manual Identity Provider Addition

Allowing users to add an identity provider manually is technically the easiest method to get this information, but it is a method that has the greatest usability drawbacks - including some that create potential security issues. Most of the information required is relatively technical and finding some way of explaining what each field is to an non-technical audience is challenging (to say the least). This especially is the case for trust anchor information. Thus this method should be considered as a power-user option only, or as a fall-back should the other methods not be applicable. Implementers may well decide not to offer the manual option due to these drawbacks.

When this method is used, careful consideration should be given to the UI presented to the user. The UI will have to ask for all of the information detailed in Section 6.2.

Trust anchors present a particularly onerous challenge for the user to enter. For this reason, many identity selectors will want to implement a leap-of-faith acquisition of the trust anchor. For these leap of faith acquisitions, the identity selector SHOULD present the user with the name of the realm that the identity selector is attempting to reach, the subject of the trust anchor certificate, details of the certification chain, and a fingerprint of the certificate. If the realm does not match the subject of the certificate, the identity selector MUST inform the user of the discrepency. The identity selector MAY reject the leap-of-faith on its own, or MAY allow the user to proceed anyway. If the user proceeds anyway, the identity selector SHOULD urge the user to reject the leap-of-faith.

The area of verification of trust anchors is very important. An Identity Selector that allows for manual addition of identity provider information SHOULD try to ensure that trust anchor information is gathered and checked in a secure a manner as possible - where users have to enter and confirm all trust anchor information, or be required to explicitly agree to an insecure configuration if this is not done properly.

6.4.1.2. Manually Triggered Automated Identity Provider Addition

One way to bypass the need for manual addition of an identity provider - and all of the usability and security issues inherent with that approach - is to provide some sort of manually triggered, but automated, addition process. One approach to accomplishing this, for example, could be for an organisation to have a section on their website where their users could visit and be given piece of data that contains much or all of the relevant identity provider information for importing into the identity selector.

Additionally, the user SHOULD be given the opportunity to:

- Supply or change the default friendly name and friendly icon for that identity provider - to allow the user to customise the identifier they use for that identity provider;

- Reject the addition of the identity provider completely - to allow the user to back out of the association process in an intuitive way.

In this case, trust anchors would be directly provided through the automated addition process to help establish the trust relationship in a secure manner.

6.4.1.3. Fully Automated Identity Provider Addition

Many organisations manage the machines of their users using enterprise management tools. Such organisations may wish to be able to automatically add a particular user’s identity provider to the identity selector on their machine/network account so that the user has to do nothing.

This represents the best usability for the user - who wouldn't actually have to do anything. However, it can only work on machines centrally managed by the organisation.

6.4.2. Identity Addition

6.4.2.1. Manual Identity Addition

Allowing users to add an identity manually is relatively easy in comparison to adding an identity provider manually. If the identity provider is already known in the identity selector, then the identity selector can construct the NAI from the identity provider and a username. Thus the manual addition of an identity in a known realm needs to prompt the user only to pick the realm, to enter the username, and to enter the credential. If the identity provider is not known to the identity selector, the identity selector will provide the user with a way to define a new one as part of the identity addition.

There are two points at which a user could manually add an identity:

- Asynchronously: the user could be allowed to, at any time, trigger a workflow of manually adding an identity. This represents the most flexible way of adding an identity since a user can perform this at any time. It does, however, also have inherent issues when it comes to verifying the newly added identity - see Section 6.6.

- Just In Time: when connecting to a service which has no mapping to an existing identity, the user could be given an option to add a new one, as well as associating with an existing one. This seems to present a better user experience when it comes to verifying the newly added identity (see Section 6.6), however, it represents a less direct method of adding an identity. Users who have not yet added the appropriate identity to their identity selector may find it difficult to understand that they must try to access a particular service in order to add an identity.

Of course, implementers could support both styles of identity addition to gain the benefits of both and give flexibility to the user.

6.4.2.2. Manually Triggered Automated Identity Addition

Much like in the case of the manually triggered automated identity provider addition Section 6.4.1.2, an identity could be added to the identity selector through a user-initiated mechanism. To follow the example in the identity provider section above, the organization could enhance the identity provider addition web service to prompt for the user part of the NAI. The web service could then generate all of the data needed for adding both the identity provider and the identity.

It is reasonable to assume that any such automated addition service is likely to be organisation specific, so that the Issuing Organisation and realm part of the NAI will be constant, as would be the trust anchor information. The user part of their NAI will have been input on the web service. The password could be provided as a part of the provided data or the identity selector could prompt the user to enter it.

If the identity provider data contained in this identity to be added conflicts with an existing identity provider known to the identity selector, the identity selector SHOULD present the discrepency to the user. The identity selector MAY reject the identity provider and identity on its own, or MAY allow the user to proceed anyway. If the identity selector allows the user to proceed anyway, the identity selector SHOULD urge the user to reject the leap-of-faith, and require the user to confirm the intent to proceed before proceeding.

Additionally, the user SHOULD be given the opportunity to:

- Supply or change the default friendly name for that identity - to allow the user to customise the identifier they use for that identity;

- Indicate whether or not the identity selector should always ask before using services with this identity - to customise the way in which the identity selector interacts with the user with this particular identity;

- Reject the addition of the identity completely - to allow the user to back out of the association process in an intuitive way.

6.4.2.3. Fully Automated Identity Addition

Section Section 6.4.1.3 introduced the concept of using enterprise management tools to add an identity provider to the identity selector. These enterprise management tools could be used to add an identity that uses the identity provider added in the above manner.

The user would not need to decipher difficult to understand data entry screens.

However, having an identity automatically provided, including its password, does have some particular usability issues. Users are used to having to provide their username and password to access remote services. When attempting to access services, authenticating to them completely transparently to the user could represent a source of confusion. User training within an organisation to explain that automated population of their identity has been enabled is the only way to counter this.

6.5. Modifying Identity Information

This process is conceptually fairly similar to adding an identity, and thus shares many of the usability issues with that process. Some particular things are discussed here.

6.5.1. Manual Modification

An identity selector may allow a user to manually modify some or all of the information associated with each identity. The obvious items that SHOULD be allowed to be changed by the user are the friendly name, the friendly icon, and the credential, or password, associated with the identity.

The identity selector should restrict other modification of the information:

- Identity Selectors SHOULD NOT allow the editing of the NAI of an identity or the trust anchor of an identity provider for items that have been added through automated means (Section 6.4.1.2, Section 6.4.1.3, Section 6.4.2.2 and Section 6.4.2.3).

- Identity Selectors MAY allow the update of the trust anchor of identity providers that have stored the trust anchor through just in time manual addition, using another just in time retrieval of the trust anchor. Any identity selector that allows this update MUST inform the user of the change in the trust anchor, and advise the user that any unexpected change should be assumed to be an attack.

- Identity Selectors SHOULD NOT allow manual modification of the password changing URL.

- Identity Selectors SHOULD NOT allow manual modification of the helpdesk URL.

6.5.2. Automated Modification

To ease usability, organisations may wish to automatically provide updates to identity provider or identity information. For example, if the user’s password changes it could automatically update the password for the identity in the user’s identity selector, or if the trust anchor information changes (e.g. if a certificate is changed) it could be automatically pushed out to all users.

6.6. Verifying an identity

An inherent by-product of the ABFAB architecture is that an identity cannot be verified during the addition process; it can only be verified while it is in use with a real service. This represents a definite usability issue no matter which method of identity addition is used (see Section 6.4):

- If the user has manually added the identity (see Section 6.4) they may have gone through the whole manual process with no errors and so believe the identity has been set up correctly. However, when they attempt to access a service, they may be given an error message, thus causing some amount of confusion.

- If the user has had the identity populated into their identity selector, then there is a much greater chance of the identity information being correct. However, if any of the information is not correct, then there is the potential for confusion as the user did not add the information in the first place.

Also, if the identity information is incorrect the user may not know where the error lies, and the error messages provided by the process may not be helpful enough to indicate the error and how to fix it (see Section 8).

6.7. Removing an Identity

This is fairly similar to adding or modifying an identity, and thus shares many of the usability issues with those processes. Some particular things are discussed here.

6.7.1. Manual Removal

Allowing the user to manually delete an identity is probably the best way to achieve the goal. Any UI should allow for this option.

6.7.2. Automated Removal

While automated removal of an identity is a way of achieving the goal without having to interact with the user, the consequence is that things may disappear from the user’s identity selector without them realising.

6.8. Storing an Identity with or without credentials

Sometimes, a user may wish to have the identity they wish to use with a service stored by the identity selector, but not the credential (e.g. password) that goes along with that Identity. The consequence of this is that when a user attempts to authenticate to a service for which an identity, but no credential, is stored, then the user would need to be prompted to manually enter the credential.

7. Considerations around Management of Service to Identity Mappings

A service to identity mapping tells the identity selector which identity should be used for a particular service. There is potentially a many-to-many association between identities and services since a user may wish to use one of their identities for many services, or more than one identity for a single service (e.g. if they have multiple roles on that service).

This potentially complex many-to-many association between identities and services is not easily comprehended by the user, and allowing the user to both manipulate it and control can be challenging. These obstacles are especially common when errors occur after an association has been made. In this scenario it is important that an identity can be disassociated with a service.

To further complicate the picture, users may wish for:

1. The identity to service mapping to be stored along with the credential, i.e. the user should always be authenticated to a particular service with a particular identity with no prompting.

2. The identity to service mapping to be stored but not the credential, i.e. the user should not be prompted to choose the identity for a particular service, but should be prompted to enter their credential for that identity.

3. The identity to service mapping to not be stored, i.e. the user should be asked which identity to use every time they authenticate to a particular service.

7.1. Associating a Service with an Identity

There needs to be a way for the user to create the service to identity association. It is advisable that this link be made only after the identity in question has authenticated with the service without any error.

There are a few ways this association could happen.

7.1.1. User-driven Manual Association

There are two ways in which manual association of an identity to a service could happen:

1. The identity selector MAY allow the user to associate a particular service with a particular identity manually, using the identity selector before they first attempt to use the service. This method is inadvisable, however, because not only might the identity in question not yet have authenticated successfully, the user would also need to know all the required technical details of that service beforehand, such as its GSS Acceptor Name.

2. On encountering a service new to the identity selector, the identity selector SHOULD pop up a dialogue box to the user asking if they would like to use an existing identity for this service (and might also allow them to create a new identity and use that).

7.1.2. Automated Rules-based Association

It would be beneficial from a usability perspective to minimise - or avoid entirely - situations where the user has to pick an identity for a particular service. This could be accomplished by having rules to describe services and their mapping to identities. Such a rule could match, for example, a particular identity for all IMAP servers, or a particular identity for all services in a given service realm. These rules could be configured as a part of the automated identity addition process described in Section 6.4.2.2 or Section 6.4.2.3.

7.1.3. Association Conflicts

The presence of rules-based associations brings with it potential conflicts in the rules. A non-exhaustive list of conflicts includes:

- One rule applies to all services of a particular type, while another rule applies to all services within a particular domain. For example, one rule applies identity A to all IMAP services, while another rule applies identity B to all services in the example.com domain.

- One rule originates from enterprise management tools as described in Section 6.4.2.3, and another rule originates from manual addition.

- The user has associated an identity with a service upon encountering the service for the first time, and later creates a rule that matches all services within that service’s realm.

Identity selectors MUST order the precedence of rules as follows:

1. Manually created rules matching specific services and realms

2. Enterprise created rules matching specific services and realms

3. Manually created rules matching any service in a single realm

4. Enterprise created rules matching any service in a single realm

5. Manually created rules matching a single service in any realm

6. Enterprise created rules matching a single service in any realm

Identity selectors SHOULD notify the user whenever a new rule will take precedence over an existing rule.

7.2. Disassociating a Service with an Identity

A user MUST be able to disassociate an identity with a service - that is, to be able to remove the mapping without having to remove the identity.

For serious authentication errors, the identity selector SHOULD prompt the user to choose whether to disassociate the identity from the service or retain the association. The prompt SHOULD explain the nature of the error.

When such a serious authentication error occurs and the identity is selected by a rules-based association (Section 7.1.2), any disassociation prompt MUST inform the user that the identity was selected by a rule. The prompt SHOULD allow the user to retain the association, or to disassociate the rule altogether. The prompt MAY include a third choice, to create an exception so that the rule does not apply to this specific service.

As of this writing, there are no authentication failures that should cause the disassociation of an identity from a service.

7.3. Listing Services and Identities

A service listing should be considered in the identity selector which is both searchable and editable by the user.

7.4. Showing the Service that is requesting Authentication

When a user is attempting to authenticate to a service for the first time, there should be some indication given to the user as to which service is requesting authentication. In many cases, the service may be obvious (where the user has started the process of attempting to authenticate to a particular service), but in other cases this may not be obvious (e.g. if an authentication attempt is triggered by a timer or a specific event), and for this scenario some indication as to the requesting service is necessary.

7.5. Showing the Identity currently in use

It would be beneficial if, when using a service, the identity currently in use could be made visible to the user while they are using a specific service. This allows the user to identify which identity is used with a particular service at a particular time (the user may have more than one identity that they could use with a particular service) - so that they can then disassociate the pairing. This is especially useful when the identity is selected without any user prompt, because of a previous association.

Implementing such a feature may be hard, however, due to the layered nature of the ABFAB transaction - the identity selector will certainly know when successful (or failed) authentications to a particular service have happened, but after that it typically plays no further part in the use of the service. Therefore, knowing that a particular service is still using a particular identity in order to indicate this to the user would be challenging.

One approach that could be used would be to display OS notifications when an identity is used. The notification could include information such as the application requesting the identity, the service receiving the identity, and the identity used. Another approach could be for the identity selector to maintain a history of identity use.

7.6. Multiple Identities for a Particular Service

An Identity Selector should be able to deal with the case where a user has multiple identities associated with a single service. For example, upon receiving a request for authentication to a service that multiple identities are configured for, ask the user which of the identities should be used in this instance.

7.7. Not using ABFAB for a Particular Service

There may be cases where a user does not wish to use ABFAB based authentication at all to a particular service, even though it is ABFAB enabled. To support this, the identity selector would have to allow the user to choose not to use ABFAB when they attempt to authenticate to a service. It would be desirable if the user could also flag that this should be remembered.

8. Handling of Errors

Errors during the ABFAB authentication process can happen at any of the many layers - they could be GSS-API errors, EAP errors, RADIUS/ RadSec errors, SAML errors, application errors, etc. ABFAB based technologies are limited in error handling by the limitations in the protocols used.

8.1. Errors in GSS-API

All GSS-API calls are necessarily instantiated from within the calling application. For this reason, when an error occurs the error is passed back to the application in order for it to deal with it. To retry, the application needs to re-initiate the GSS-API call. Unless the application has been written to deal with this properly, this process can be very tedious for a user and cause them opt out of what they are trying to accomplish. In addition to this, the application may not display the error messages to the user. Even when the application does display the errors, the messages themselves may not be useful enough for the user to decipher what has gone wrong.

Two extensions to GSS-API are suggested for the consideration of the kitten working group:

- GSS-API should provide a method for applications to invoke to indicate that the application has displayed the last error to the user.

- GSS-API should provide a method for applications to invoke to indicate that the authentication succeeded, but is insufficient for the task at hand and needs to be retried.

8.1.1. Log of Errors

The Identity Selector can improve the general GSS-API error reporting experience by displaying a list of errors experienced by ABFAB applications. When an application error occurs, the EAP mechanism MAY record that error. If the mechanism records these errors, the Identity Selector MAY display these errors to the user. Thus, the user will have a single place to go to view all of the errors that a user experiences across all applications. Therefore, an Identity Selector that implements an error display SHOULD present the user with the context of the error, including the calling application and the time.

8.2. Examples of errors

To give an idea of the range of errors that might be seen, consider the following non-exhaustive set of potential errors.

Identity Association/Verification Errors:

- The credentials presented to the IdP were not able to be verified - e.g. wrong username/password.

- The Trust Anchor for the IdP was invalid.

Service Errors:

- The IdP recognizes the client, but decides not to authorize it for this service.

- The EAP session succeeds, but the RADIUS system sends access-reject to the Relying Party

- The RADIUS system succeeds, but the Relying Party rejects the session. For instance, the SAML part of the session could contain an error that causes the Relying Party to reject the client.

- The Identity might have been successfully authenticated, but the user might not have authorisation to use the service or privilege levels within the service they are attempting to use. For instance, the Identity could authorise the use of an operating system as an unprivileged user, which would prevent the user’s goal of managing the hard drives.

Other Errors:

- The IdP didn’t respond to the Service.

- The IdP didn’t respond to the Client.

- Network errors.

- Timing errors.

9. Handling of Successes

It is of course hoped that the identity selector will have to occasionally handle successes as well as errors. This section has some brief discussion about some areas you might want to think about.

9.1. Reporting Authentication Success on First Use of Identity

The first time an identity is used with a service, it would be good practice to visually indicate in some way that the process has been successful, in order that the user understands what is happening and is then prepared for future authentication attempts.

9.2. Reporting Authentication Success

On an on-going basis you may or may not wish to indicate visually to the user a successful authentication to a service. This relates to Section 7.5.

10. Other Considerations

This section briefly discusses other considerations that you might want to think about that don’t fit in any of the other categories.

10.1. Identity Selector Taking Focus

When an ABFAB authentication request is triggered, and where it needs input from the user, the Identity Selector should take focus in some way so that it is clear to the user that they need to do something to proceed.

10.2. Import/Export of Credentials

For various reasons, an identity selector implementation might want to include functionality that allows for the export/import of identities and service to identity mappings. This could be for backup purposes, to allow a degree of mobility between identity selector instances, etc.

If providing this functionality, it would be advisable that the credential store that is the result of the export should be secure - encrypted and password protected - given the nature of the information.

11. Security Considerations

Most security considerations are ones relevant to the use of GSS-EAP and are detailed in [I-D.ietf-abfab-arch]. There are, however, a few specific sets of security considerations related to the UI implementation.

First, as discussed earlier, the Identity Selector should use a Trust Anchor to authenticate the IdP before it sends the users credentials to it. Having no Trust Anchor information at all, or an incorrect Trust Anchor, can enable the possibility of someone spoofing the IdP and harvesting credentials sent to it. So, how this Trust Anchor is configured and managed can have major security implications:

- The most secure way for a Trust Anchor to be configured is to have it provisioned alongside the other identity information in an enterprise provisioning scenario. This allows for the correct Trust Anchor to be configured with no user input required. However, thought needs to be given to Trust Anchor expiry and consequent requirement for regular reprovisioning of identity information.

- Another way that is potentially secure would be to allow the user to discover the Trust Anchor information out of band and manually input this information into the Identity Selector. This is only secure, however, for those users who understand what they’re doing in this scenario; pragmatically, this is unlikely to be the case for many users so is not a recommended approach for the average user.

- A pragmatic approach would be leap of faith, whereby no Trust Anchor information is initially provisioned, and the first time the Identity Selector connects to the IdP it remembers the Trust Anchor information for future use. This doesn’t mitigate against spoofing of an IdP in the first instance, but would enable mitigation against it for all future connections.

- Finally, there may be interesting ways to leverage technologies such as DANE [RFC6698] to store the Trust Anchor for an IdP in DNS.

Secondly, the storage of the user’s credentials by the Identity Selector should be done in a secure manner to mitigate against people taking unauthorised control of the device being able to gather these credentials. Use of a secure credential storage mechanism, such as the GNOME Keyring on Linux, or Keychain on the Mac, are recommended.

12. Privacy Considerations

Since the ABFAB system facilitates the sharing of identifying information about a user, the undesired sharing of information is a real concern. Most of the privacy considerations lie outside the scope of the Identity Selector UI, which neither controls nor sees which attributes of an identity will be shared with a service. In essence, the only control that the Identity Selector has is whether or not a given identity will be shared with the service.

However, the selection of identity does warrant privacy considerations. Any automated choice of identity for a service will share information, potentially inappropriately. Examples of this include:

- Rules that apply to a service across all realms will cause an identity choice, even for realms the user would actually prefer to avoid interacting with at all.

- Storing a default for a particular service and realm will cause the identity to be selected in that situation going forward, even if the situation or application does not warrant that. For instance, a web browser in privacy mode ideally should not know of any saved identity association choices.

Even appropriate choices of sharing an identity with a service leaks information about the user. The desired service and the identity provider must communicate with each other to perform an authentication. Even if the authentication fails, the service will know the realm of the user credential, and the Identity Provider will know the realm, and maybe the service, that the user tried to access. For services with fallback authentication mechanisms, the system may try and fail to authenticate the user, thus sharing the realm information noted above, without the user being aware this has happened.

13. IANA Considerations

This document does not require actions by IANA.

14. Contributors

The following individuals made important contributions to the text of this document: Sam Hartman (Painless Security LLC), and Maria Turk (Codethink Ltd).

15. Acknowledgements

Thanks to Jim Schaad, Stefan Winter, David Chadwick, Kevin Wasserman, Alejandro Perez-Mendez, Ken Klingenstein, and Dave Crocker for feedback and suggestions.

16. References

16.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The Network Access Identifier", RFC 4282, DOI 10.17487/RFC4282, December 2005, <http://www.rfc-editor.org/info/rfc4282>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <http://www.rfc-editor.org/info/rfc6698>.

[I-D.ietf-abfab-arch] Howlett, J., Hartman, S., Tschofenig, H., Lear, E., and J. Schaad, "Application Bridging for Federated Access Beyond Web (ABFAB) Architecture", draft-ietf-abfab-arch-12 (work in progress), February 2014.

[I-D.ietf-abfab-usecases] Smith, R., "Application Bridging for Federated Access Beyond web (ABFAB) Use Cases", draft-ietf-abfab-usecases-05 (work in progress), September 2012.

16.2. Informative References

[MS-CS] Brown, K., "The InfoCard Identity Revolution", July 2006, <https://technet.microsoft.com/en-us/magazine/2006.07.infocard.aspx>.

Appendix A. Change Log

Note to RFC Editor: if this document does not obsolete an existing RFC, please remove this appendix before publication as an RFC.

IETF draft -03 to ietf draft -04

1. Document service errors.

2. Document GSS error handling, including a request for a couple of new GSS methods, and maintaining a log of all GSS errors for later viewing.

IETF draft -02 to ietf draft -03

1. Tidying up language throughout.

2. Added the idea of an identity provider object within the identity selector, and moved the trust anchor property from the identity to the identity provider.

3. Added restrictions on manual modification of automatically added identities and identity providers.

4. Added precedence between identity association rules.

5. Incorporated many comments from the mailing list.

6. Added privacy considerations section.

IETF draft -01 to ietf draft -02

1. Tidying up language throughout.

2. Finished remaining TODOs - largely in the error handling section.

3. Added security considerations section.

IETF draft -00 to ietf draft -01

1. Tidying up language throughout

2. Doing some of the TODOs

3. Added language that tries to explain that this document is not a substitute for good HCI/UX design.

4. Changed terminology slightly to avoid confusion between an identity selector "mechanism" and a GSS-API mechanism.

5. Added a caveat about the potential for the UI to show the identity currently in use for a particular service.

6. Added a requirement that the identity selector must not store the same NAI for multiple identities.

7. Stopped talking about "provisioning" after saying that I wouldn’t talk about "provisioning".

Draft -04 to ietf draft -00

1. Adding brief discussion of identities vs identifiers (Ken).

2. Changing assumption about credentials having a password in favour of more generic text for other auth types.

3. Adding discussion of storage of identity information.

4. Added sections on dealing with multiple identities per service, remembering credentials, remembering not to use ABFAB.

5. Added small section on ID selector needing to take focus in some way.

Draft -03 to draft -04

1. Addressing various comments from Jim and Stefan.

Draft -02 to draft -03

1. Bumping version to keep it alive.

Draft -01 to draft -02

1. Completed the major consideration sections, lots of rewording throughout.

Draft -00 to draft -01

1. None, republishing to refresh the document. Other than adding this comment...

Appendix B. Open Issues

Note to RFC Editor: please remove this appendix before publication as an RFC.

Authors' Addresses

Dr. Rhys Smith
Cardiff University
39-41 Park Place
Cardiff CF10 3BB
United Kingdom

Phone: +44 29 2087 0126
EMail: smith@cardiff.ac.uk

Mark Donnelly
Painless Security
14 Summer Street
Suite 202
Malden, Massachusetts 02176
United States

EMail: mark@painless-security.com